N73-19267

FINAL TECHNICAL REPORT September 30, 1972

WILLIAM MARSH RICE UNIVERSITY

DEPARTMENT OF ELECTRICAL ENGINEERING

LABORATORY FOR COMPUTER SCIENCE AND ENGINEERING

FEASIBILITY INVESTIGATION OF A CELLULARLY ORGANIZED DATA PROCESSOR

JPL CONTRACT 953246
Subcontract under NASA CONTRACT NAS7-100
Task Order No. RD-95

by

Robert C. Minnick, Principal Investigator
J. Robert Jump, Co-Principal Investigator
Robert G. Arnold, Student Assistant
John M. Beirne, Student Assistant

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration under Contract NAS7-100.

I. INTRODUCTION

The purpose of this project has been to study the application of cellular arrays to NASA missions. Cellular arrays are iterative logical and memory structures which can be programmed to accomplish a wide variety of logical tasks. Used in long-duration space missions, for example, spare cellular arrays can be remotely programmed to replace faulty logical subsystems as the need arises.

While cellular arrays have been studied for a considerable period of time*, only a modest amount of effort has been given to the practical problems of implementation. A major purpose of this project has been to address these practical problems.

The approach that has been taken is to study three different cellular arrays, and to realize three different logical designs for each. The cellular arrays that were chosen are

1. The microprogrammed array [2].
2. The cobweb array [3].
3. The programmable array [4].

Array 1 has complex cells: each has as much logic as a one-bit arithmetic unit. Array 2 has somewhat less complex cells, while array 3 has simple NOR elements as cells. Thus, the arrays that were chosen for study cover a wide range of types.

*An extensive bibliography is contained in [1].

Similiarly, the three logical designs were chosen to represent a broad spectrum of NASA tasks. They are

1. A seven-bit serial multiplier.
2. A 31-bit sequence detector.
3. A 31-bit feedback shift-register decoder.

All these cellular arrays were implemented using two different techniques: a conventional random logic configuration, and a <u>programmable logic array</u> (PLA) configuration. For this PLA technique of Texas Instruments [5], combinational logical functions are realized in a read-only memory (ROM) where the word-selection logic can be programmed in the same way as the storage array. The structure consists of two arrays: the first array is used to realize the product terms and the other is used to realize the sum terms of a set of logical functions in sum-of-product form.

There are two potential advantages of the PLA and similar techniques over the random logic technique. First, the initial cost of customizing an array is less since only one mask must be modified and the complex problem of wire layout is avoided completely. Second, the PLA technique frequently results in less chip area than an equivalent random logic realization.

II. MICROPROGRAMMED ARRAY

A. General Description*

The interconnection structure of the microprogrammed array is

*Portions of this section are based on [2].

illustrated in Fig. 1, where all intra-cell lines propagate binary signals in the indicated directions. All cells in the array are identical logic networks with the form indicated in Fig. 2. Q_1, Q_2, Q_3 and D are storage elements which determine the function performed by the cell. The operation of loading a microprogram into the array is equivalent to loading these storage elements, in every cell, to implement the desired microinstructions in the rows of the array. The five output signals are combinational logical functions of the storage elements and the five cell-input signals.

The three storage elements Q_1, Q_2 and Q_3 determine a cell's _type_. In Table I the eight different cell types and behavior are specified by means of the logical equations for the cell output terminals. In the first three types, the _data flip-flop_ D is treated as a variable which may change during execution. D can not be changed during execution if the cell is programmed to one of the last five types in the table. For the last four, it is used to modify the cell type.

Although the functional behavior of the array is completely defined by Fig. 1 and Table I, some additional comments on the use of the different cell types are in order. For this purpose, the cell types are classified into three categories: _computation-mode_ (types 1, 2 and 3), _path-mode_ (types 4 and 5), and _control-mode_ (types 6, 7 and 8). The descriptive symbols of Fig. 3 are used to represent the different cell types. In order to simplify array diagrams, only the X, Y and S lines are shown.

In order to explain the operation of the cell, it is also convenient to classify the intra-cell lines of the array as control lines (G, L and S) and data lines (X, Y and S). The data lines carry the data signals generated and transformed by the computation-mode and path-mode cells of an array. The control lines carry signals, generated by control-mode cells, which control the operation of the computation-mode and path-mode cells. The S line serves as a control line for computation-mode cells and a data line for path-mode cells. This is indicated by the dot on the S input line of the computation-mode cells.

The control lines G and L control the output signal $\hat{Y}$ of computation and path-mode cells. Outputs $\hat{X}$ and $\hat{S}$ are not a function of the value of G or L. If either G or L has value 0, then $\hat{Y}=Y$. Hence information passes through the cell, without modification, in the vertical direction. In this case the cell is said to be inactive. For some microprograms, the control signal L of some cells may be fixed at 0. In this case the modified symbols shown in Fig. 4 are frequently used to emphasize that $\hat{Y}=Y$. If both inputs G and L have value 1 the cell is said to be active. In this case, $\hat{Y}$ is determined by the cell type. Since both of these control lines are effectively bussed through the cell, several cells in the same row can be activated simultaneously.

The control line G is used as an "enable" line and control line S as a "clock" line for the data flip-flop in computation-mode cells. Control line S is also bussed through computation-mode cells so that data can be simultaneously set into several cells in the same row.

Computation-mode cells are the only ones whose data flip-flop can be altered during the execution of a microprogram. Hence S serves as a data line for path-mode cells.

An ADD cell realizes the function of a full adder with inputs X, Y and D, carry output $\hat{X}$ and sum output $\hat{Y}$. Thus its primary use is in the realization of arithmetic microinstructions. It can also be used to produce two-variable logical functions of inputs X and Y. With D set to 0, $\hat{X}=X\cdot Y$ and $\hat{Y}=X\oplus Y$. With D set to 1, $\hat{X}=X\vee Y$ and $\hat{Y}=\overline{X\oplus Y}$.

The horizontal logic of a COMP cell can be used to test for equality of two words. Indeed, if the signal $\hat{X}$, produced by the leftmost cell of a row of COMP cells, is 1, then the word on the Y inputs to the row must be identical to the word stored in the D flip-flops of the row. This cell function is useful in implementing the control structure of a microprogram and in realizing an arbitrary product term of Boolean variables. The vertical logic of a COMP cell provides the logical complement of input Y at output $\hat{Y}$.

The primary function of a REGISTER cell is to store the operands of a computation. Both output signals $\hat{X}$ and $\hat{Y}$ equal the value of D. Hence it can also be used to produce the logical constants 0 and 1 at either the X or Y input of an adjacent cell. The L-SHIFT, R-SHIFT and NULL cells are used to form data paths between cells in the obvious way.

After a thorough study of the microprogrammed array as it was first reported [2], a modification was made in order to increase its flexibility. This modified array can be used effectively to perform each of the three tasks of the project by reprogramming. The original version of the cell could only propagate information from top to bottom in the vertical direction. Thus it was sometimes necessary to route information from the output terminals along the bottom of the array to the input terminals at the top. This required additional switching logic external to the array. The new version of the cell has an additional data path in the vertical direction which is used to propagate information from the bottom to the top of the array. A new cell type has also been added to perform the operation of reversing information flow in the vertical direction.

The functional behavior of the modified cell is specified by the equations in Table II and the diagram in Fig. 5. The new signals are Z and $\hat{Z}$ and the new cell type is REVERSE. The signals P, $\hat{P}$, C and $\hat{C}$, used to load specification bits initially are also shown in Fig. 5.

B. Cell Realization

In order to realize the microprogrammed cell using the PLA technique, the cell equations must be put into a sum-of-product form. Standard two-level minimization techniques can be used effectively for this cell. The equations for the cell are given in Table III. These equations along with the diagram in Fig. 6 completely specify the PLA implementation of the microprogrammed cell.

To obtain a random logic realization of the microprogrammed cell, the cell equations were modified to reduce the number of gates. Two, three and four-input gates were allowed and the restriction of two-level logic was removed. The resulting equations are given in Table

C. Loading the Specification Bits

Loading the specification bits of the microprogrammed cell is complicated since one of them (the D bit) can be changed during exec of the array. Hence the cell has been designed so that it has two p of operation. During the loading phase, the D flip-flop and the thre fixed specification flip-flops Q_1, Q_2, Q_3 are configured into a four-bit shift register so that a column of cells can be loaded serially. During the execution phase, the input to the D flip-flop comes from the cell logic instead of the flip-flop Q_3. The phase is determined the control signal P which is asserted during the loading phase. In order to minimize the number of terminals of a cell, the Y and $\hat{Y}$ terminals of a cell are also used as input and output for the shift register during the loading phase. The organization of the specification flip-flops is shown in Fig. 6.

D. Sample Designs

1. Multiplier

Using the microprogrammed cell, it is feasible to design a multiplier having either a parallel or a serial word organization. Hence, both designs have been completed. The seven-bit serial multiplier is shown in Fig. 7. It requires a 6 x 23-cell array, or 138 cells.

In reference to Fig. 7, both the multiplier and the multiplican are loaded into the array.

B. Cell Realization

In Fig. 17, a NAND-NOR realization is shown for one cobweb cell. By comparing Figs. 16 and 17 it can be verified that by setting the specification bits S_1, S_2, S_3 and S_4 appropriately, the cell output, z, is one of the specified functions of the cell input, x and y. The R-S flip-flop in Fig. 17 is enclosed in dotted lines.

Since each cobweb cell has five inputs, as is shown on Fig. 15, but produces an output that is dependent on only one or two of these, some additional input selection logic to that shown in Fig. 17 is needed. While the original cobweb cell used cutpoints for this purpose, it is possible to accomplish the same results using electronic techniques. One attractive circuit for the input selection logic is shown in Fig. 18. This circuit uses open-collector TTL NAND gates with a single pull-up resistor per line or buss. If a specification bit for one of the open-collector NAND pairs is at V_{cc}, then the output is the complement of the input. On the other hand, if a specification bit is at ground, that NAND pair disconnects the input from the output. With the circuit shown in Fig. 18, one has the advantage of a wired-OR for the inputs, and furthermore the busses can be used for bilateral jumpered connections.

The complete cobweb cell realization is shown as Fig. 19. This incorporates the logic of Figs. 17 and 18. The R-S flip-flop of this cell can readily be changed to one of another type. For instance, if one elects to use a master-slave D flip-flop, the realization of Fig. 20

results. A block form of the cobweb cell is shown as Fig. 21.

The PLA implementation of the cobweb cell uses two arrays. One is used to realize the selection logic while the other realizes the cell function logic. This was done to minimize logic since the cobweb cell is not well suited for two-level logical realization. This implementation also differs from the random logical implementation in that the number of specification bits has been reduced by coding the possible specification words. This was done in this case since the realization of a decoder in the PLA array uses less area than the flip-flops that are eliminated. The PLA implementation is given in Fig. 22 and Table V. It should be noted that the R-S flip-flop in the cobweb cell is realized by means of the feedback line in the second PLA array.

C. Loading the Specification Bits

The specification bits for the cobweb cell can be stored in various ways. For the purpose of this study it is assumed that they are stored in a shift register within the cell. These shift registers are arranged so that the specification bits of a column of cells can be loaded serially.

In the cobweb cell, this is straightforward since the specification bits are not changed once they are loaded. Hence, unlike the microprogrammed arrays, the operation of the shift register is independent of the function of the cell. The shift register is controlled by two additional inputs and outputs, L and C, as shown in the PLA realization of Fig. 22. Signals L and $\hat{L}$ are the data input and output signals for the shift register and C is the clock.

D. Sample Designs

1. Multiplier

A seven-bit serial multiplier which uses the cobweb array is shown as Fig. 23. It consists of four seven-bit shift registers which store the multiplier, the multiplicand and the two halves of the product. The locations of these shift registers in Fig. 23 are indicated by the overlay of Fig. 24. The multiplicand register contains one extra bit that is initially set to zero. T is a signal that is a one during the first and each following eighth step and zero during all others. The five control functions for the multiplier are given in terms of the three control lines α,β,γ and are shown in Table VI.

2. Sequence Detector

A cobweb-array realization for a 31-bit sequence detector is shown as Fig. 25. In this detector the inverted output of each position in the storage shift register is exclusive-OR'ed with the corresponding position in the code. The AND of the exclusive-OR output is the desired result.

In order to illustrate the results when cobweb arrays designed according to Fig. 20 are used rather than those designed according to Fig. 19, a seven-bit sequence detector using each of these arrays appears as Fig. 26. The standard cell design requires a 5 x 28 array of which 23 cells are not used. The modified cell design requires a 3 x 13 array of which one cell is not used. Furthermore the standard array uses a four-phase clock while the modified array requires only a single-phase clock.

3. Feedback Shift Register Decoder

A 31-bit feedback shift register decoder is shown as Fig. 27. It requires a 29 x 14-cell array.

IV. PROGRAMMABLE ARRAY

A. General Description*

The programmable array is based on an earlier array of simple NAND cells [6]. The original array, which was due to Spandorfer and Murphy, is shown as Fig. 28. Each cell in this array is a one, two or three-input, single-output NOR or NAND. In this simple array the logic of each cell is fixed, while a subset of the interconnection array is chosen by selectively opening the cutpoints shown as small circles and triangles. For ease in drawing, the single output of some cells (like cell 1,2) is shown connected at several points on the periphery of the cell. This array connects each of n variables x_i or x_i' or neither to cells $(2i-1,2j)$, $1 \leq i$, $j \leq n$, through the cutting of one or both of the arcs marked with small circles. Then by cutting all arcs marked with small triangles except those immediately above the bottom row, one obtains the desired switching function F in a

*Portions of this section are based on [4]

NOR-NOR form.

An inspection of Fig. 28 shows that the interconnection structure repeats with a cycle length of two in both directions. Hence, it is reasonable to define a 2 x 2 subarray as a single cell; this is shown as Fig. 29. This new cell of Fig. 29 has easily determined logical properties. It is the version of the programmable cell [4] which has appeared in the literature.

As originally reported, the programmammable cell had two specification bits which allowed it to be set to one of four conditions as follows:

Input x is connected.

Input x' is connected.

Neither input is connected.

Collector row is connected.

After a thorough study of the applicability of this array to practical logical designs, it was determined that a modified version of this cell would lead to more efficient results. In this modified version, a turning condition was added to the collector row connection such that the y input was connected to the X output (see Fig. 29).

For this revised cell the logical equations are

$$X=(S_1S_2)'x+S_1S_2(z'+y)$$

$$Y=z'+S_1y+S_1'S_2'xy+S_1'S_2x'y$$

$$Z=y'+S_1'+S_2'$$

where S_1 and S_2 are two specification bits. The revised cell is drawn in Fig. 30. The numbers in the cell of part (b) of that figure represent the values for the specification bits S_1, S_2, expressed in decimal form.

B. Cell Realization

A simple realization for this programmable cell would be in terms of a uniform array of NAND elements. Therefore, following [4], a 5 x 5 array of NAND's which realizes the cell of Fig. 30 is drawn as Fig. 31. This cell also can be realized using PLA and similar techniques.

It is seen from Fig. 31 that the following count of NAND elements is used:

5	1-input
8	2-input
7	3-input
4	4-input

C. Loading the Specification Bits

In Fig. 31, the two flip-flops which represent S_1 and S_2 are set and reset as follows:

Set S_1:	$L_aL_bC_1'C_2$
Reset S_1:	$L_aL_bC_1'C_2'$
Set S_2:	$L_aL_bC_1C_2$
Reset S_2	$L_aL_bC_1C_2'$

The L_a and L_b lines are used in a coordinate-access method which allows one cell like that shown in Fig. 31 to be selected from an array of cells. Such an array is shown as Fig. 32. In Fig. 32 the signals C_1 and C_2 result in the appropriate specification bit being set or reset according to the above equations.

D. Sample Designs

1. Multiplier

In order to find practical realizations for this simple microprogrammed cellular array, it is necessary to perform any storage externally to the array. It should be noted for the other two cellular arrays that were studied that this was not a requirement.

For this reason, the design of a seven-bit multiplier reduces to the design of a seven-bit parallel adder combined with the necessary controls. One bit of such a gated full adder is shown as Fig. 33; seven such arrays laid out horizontially in a 6 x 63 array yield the complete adder. The remaining registers, as well as the load and shift controls are outside the array.

2. Sequence Detector

A 31-bit sequence detector requires a 5 x 124 array of programmable cells. One of the 31 stages for this detector is shown as Fig. 34. As in the case for the multiplier, the two registers for this sequence detector circuit are external to the cellular array.

3. Feedback Shift Register Decoder

In order to realize a feedback shift register with a programmable cellular array, the actual register is formed externally, and only the logic of the feedback is determined in the array. From Fig. 30 it is seen that a cascade of 0, 1 and 2 cells, with the z input having the value of 1, can be found for any product of literals on the x inputs. Furthermore, the 3-cell can be used to form the disjunction of these conjunctions. For example, it will be supposed that the following feedback logic is to be found:

$$F=(x_1 \oplus x_3 \oplus x_5)' = (x_1' x_3' x_5 + x_1' x_3 x_5' + x_1 x_3' x_5' + x_1 x_3 x_5)'$$

A programmable array which accomplished this is shown as Fig. 35. Clearly, any other feedback logic can be found in a similar array. For instance, a 31-stage feedback shift register which uses 16 terms in the feedback equation requires a 32 x 16-cell array plus the external shift register.

V. COMPARISONS

A. Assumptions

Estimates of the chip area required for the cell implementations described in Section II, III and IV have been calculated. In order to make these calculations, the area requirements for gates, random logic wiring, PLA arrays and flip-flops given in [5] have been used. Although these figures may vary considerably with different technology and fabrication techniques, it is felt that they will give valid relative comparisons of the different implementations considered.

For the area of a MOS gate, the following formula was used:

$$(\text{AREA})_G = (12 \text{ mils}) \times (\text{number of pins/gate}) \times (1.2 \text{ mils/pin}).$$

This figure does not include the area needed to interconnect the gates. It was assumed that from 50% to 80% the circuit area was used for interconnection leads.

A PLA array uses from one to two square mils per bit. Thus the area of an array is given by the following formula.

$(\text{AREA})_A = (1 \text{ or } 2 \text{ mils}^2) \times (\text{number of product terms}) \times (\text{number of sum terms} + 2\,(\text{number of variables}))$.

The area required for the specification bit shift register was assumed to be from 100 to 300 mils^2 per stage.

B. Area Calculations

The results of the area calculations are given in Tables VII and VIII. In Table VII the area required for PLA implementation is presented for four different assumptions. The area required per shift register stage was assumed to be either 100 mils^2 or 300 mils^2 and the area required per PLA bit was assumed to be either 1 mil^2 or 2 mil^2. The are requirements for random logic implementation were also computed for four different assumptions as shown in Table VIII.

C. Discussion

The numbers in Tables VII and VIII provide a measure of the relative complexity of the microprogrammed cell, the cobweb cell and the programmable cell. Some observations on these results can be made:

1. Most of the trends in the table can be predicted from the fact that the area used for cell logic dominates for the microprogrammed cell, while the shift register dominates for the cobweb cell.
2. In all cases considered, the PLA implementation used less

area than the random implementation. This difference was greatest and is nearly the same ratio in the case of the microprogrammed and the programmable cells, but the random logic version of the cobweb cell still requires approximately twice as much area as the PLA version.

3. Comparing the random logic versions of the three cells, the cobweb cell requires from 40% to 80% as much area as the microprogrammed cell, while the programmable cell requires from 21% to 24% as much area as the microprogrammed cell.
4. Comparing the PLA versions of the two cells, the cobweb cell requires from 60% to 120% as much area as the microprogrammed cell, while the programmable cell requires from 25% to 30% as much area as the microprogrammed cell.

VI. SUMMARY AND CONCLUSIONS

In Table IX and X the nine combinations of array types and designs are summarized for PLA and random logic implementation. For row 1 of these tables, the serial version of the microprogrammed-cell multiplier was used. In row 3, the programmable realization includes 22 bits of storage that is external to the cellular array. In row 6, the programmable array includes 62 bits of external storage. For row 7, a linear feedback shift register having a serial load is assumed. Sixteen terms are assumed in the feedback equations on rows 7 and 9. On row 9 an external 31-bit shift register is assumed as well.

While Tables IX and X gives some measure of comparisons, it should be emphasized that because of the essential differences among the arrays that were studied, a direct numerical comparison of chip sizes may be misleading. In particular, the sizes shown in the table for the programmable array are low because all storage except for the specification bits appears outside the array. Thus, while the micro-programmed and the cobweb realizations each are in terms of single arrays, the programmable realizations are in terms of two or more arrays when the external registers are counted.

It should also be observed that in some cases the ground rules differed slightly for the comparative designs because they were accomplished by different persons at different times.

With this caveat in mind, some conclusions can be drawn from the results in Tables IX and X. First, it is clear that as the cell is made more complex, the efficiency of the design increases. Second, the programmable array could be greatly improved by the addition of one or more storage bits per cell. Third, the cobweb array could be improved by using a more sophisticated flip-flop, and perhaps by increasing its number of storage bits per cell. Fourth, since all three cellular arrays that were studied were modified in the light of practical design tasks, it is expected that further study would lead to even more improvements to these and other cellular arrays. Fifth, PLA or similar techniques appear to be better suited for the realizations of all the cellular arrays than is random logic.

REFERENCES

1. Minnick, R.C., "A Survey of Microcellular Research," Journal of the Association for Computing Machinery, Vol. 14, No. 2, pp. 203-241, April 1967.

2. Jump, J.R. and Fritchie, D.R., "Microprogrammed Arrays," IEEE Trans. on Computers, Vol. C21, No. 9, pp. 974-984, September 1972.

3. Minnick, R.C., "Cobweb Cellular Arrays," Proceedings of the 1965 Fall Joint Computer Conference, pp. 327-341, AFIPS, Vol. 27, Spartan Books, 1965.

4. Minnick, R.C., "A Programmable Cellular Array," Proceedings of the 1971 IEEE International Computer Group Conference, pp. 25-26, Institute of Electrical and Electronic Engineers, Inc., 71C41-C (September 22-24, 1971).

5. Andres, K., "MOS Programmable Logic Arrays," Technical Report, Texas Instruments, Inc., Dallas, Texas.

6. Spandorfer, L.M. and Murphy, "Synthesis of Logic Functions on an Array of Integrated Circuits," final report for UNIVAC Project 4645, prepared for the AFCRL on Contract 19(628)2907 (November 30, 1965).

CELL TYPE \ OUTPUT TERMINAL	$\hat{X}$	$\hat{Y}$	$\hat{S}$	$\hat{L}$	$\hat{G}$	D_{set}	D_{clear}
1 - ADD	$XY \vee XD \vee YD$	$(\bar{G} \vee \bar{L})Y \vee GL(X \oplus Y \oplus D)$	S	L	G	GSY	$GS\bar{Y}$
2 - COMP	$X(Y \oplus \bar{D})$	$(\bar{G} \vee \bar{L})Y \vee GL\bar{Y}$	S	L	G	GSY	$GS\bar{Y}$
3 - REGISTER	D	$(\bar{G} \vee \bar{L})Y \vee GLD$	S	L	G	GSY	$GS\bar{Y}$
4 - NULL	X	Y	S	L	G	0	0
5 - L-SHIFT (D=0) R-SHIFT (D=1)	$\bar{D}Y \vee DX$	$(\bar{G} \vee \bar{L})Y \vee GL(\bar{D}X \vee DS)$	$\bar{D}S \vee DY$	L	G	0	0
6 - LOCAL (D=0) STORE (D=1)	Y	Y	$S \vee DG$	$L \vee \bar{D}G$	G	0	0
7 - BOUNDARY (D=0) BEND (D=1)	$D \vee X$	$\bar{G}Y \vee G(\bar{D}Y \vee DX)$	$\bar{D}SY \vee DS$	L	$\bar{D}(L \vee S)$	0	0
8 - BLOCK (D=0) PARTITION (D=1)	DX	$\bar{G}Y \vee G(\bar{D}X \vee DY$	S	Y	G	0	0

TABLE I. CELL EQUATIONS FOR THE MICROPROGRAMMED ARRAY

OUTPUT TERMINAL / CELL TYPE	$\hat{X}$	$\hat{Y}$	$\hat{Z}$	$\hat{S}$	$\hat{L}$	$\hat{G}$	D_{set}	D_{clear}
1 - ADD	$XYvXDvYD$	$(\overline{G}v\overline{L})YvGL(X\oplus Y\oplus D)$	Z	S	L	G	GSY	$GS\overline{Y}$
2 - COMP	$X(Y\oplus\overline{D})$	$(\overline{G}v\overline{L})YvGL\overline{Y}$	Z	S	L	G	GSY	$GS\overline{Y}$
3 - NULL (D=0) / REVERSE (D=1)	X	$\overline{D}YvDZ$	$\overline{D}ZvDY$	S	L	G	0	0
4 - REGISTER	D	$(\overline{G}v\overline{L})YvGLD$	Z	S	L	G	GSY	$GS\overline{Y}$
5 - L-SHIFT (D=0) / R-SHIFT (D=1)	$\overline{D}YvDX$	$(\overline{G}v\overline{L})YvGL(\overline{D}XvDS)$	Z	$\overline{D}SvDY$	L	G	0	0
6 - BOUNDARY (D=0) / BEND (D=1)	DvX	$\overline{G}YvG(\overline{D}YvDX)$	Z	$\overline{D}SYvDS$	L	$\overline{D}(LvS)$	0	0
7 - LOCAL (D=0) / STORE (D=1)	Y	Y	Z	$SvDG$	$Lv\overline{D}G$	G	0	0
8 - BLOCK (D=0) / PARTITION (D=1)	DX	$\overline{G}YvG(\overline{D}XvDY)$	Z	S	Y	G	0	0

TABLE II. MODIFIED CELL EQUATIONS FOR THE MICROPROGRAMMED ARRAY

$\hat{X} = \overline{Q}_1Q_2Q_3D + \overline{Q}_2\overline{Q}_3D\,X + \overline{Q}_1Q_2\overline{Q}_3X + Q_1Q_2\overline{Q}_3Y + Q_1\overline{Q}_2Q_3X + Q_1\overline{Q}_2Q_3D + Q_2Q_3D\,X + Q_1\overline{Q}_3\overline{D}\,Y + \overline{Q}_1\overline{Q}_3X\,Y + \overline{Q}_1\overline{Q}_2\overline{Q}_3D\,Y + \overline{Q}_1Q_3D\,X\,Y + \overline{Q}_2Q_3\overline{D}\,X\,\overline{Y}$
$\hat{Y} = \overline{P}\,Q_3Y\,\overline{G} + \overline{P}\,\overline{Q}_2Y\,\overline{G} + \overline{P}\,\overline{Q}_1\overline{Q}_2Y\,\overline{L} + \overline{P}\,\overline{Q}_2\overline{Q}_3\,Y\,\overline{L} + \overline{P}\,\overline{Q}_1Q_3Y\,\overline{L} + \overline{P}\,Q_1Q_2\overline{Q}_3Y + \overline{P}\,Q_1Q_2Y\,D + \overline{P}\,Q_2\overline{Q}_3Y\,\overline{D} + \overline{P}\,Q_1\overline{Q}_2Q_3Y\,\overline{D} + \overline{P}\,\overline{Q}_1\overline{Q}_2Q_3\overline{Y}\,G\,L + \overline{P}\,\overline{Q}_1\overline{Q}_2X\,\overline{Y}\,\overline{D}\,G\,L + \overline{P}\,\overline{Q}_1\overline{Q}_2\overline{X}\,\overline{Y}\,D\,G\,L + \overline{P}\,\overline{Q}_1\overline{Q}_2\overline{Q}_3X\,Y\,D + \overline{P}\,\overline{Q}_1\overline{Q}_3\overline{X}\,Y\,\overline{D} + \overline{P}\,\overline{Q}_1Q_2Q_3D\,G\,L + \overline{P}\,Q_1\overline{Q}_2Q_3X\,D\,G + \overline{P}\,Q_1Q_2Q_3X\,\overline{D}\,G + \overline{P}\,Q_1\overline{Q}_2\overline{Q}_3X\,\overline{D}\,G\,L + \overline{P}\,Q_1\overline{Q}_2\overline{Q}_3D\,G\,L\,S + \overline{P}\,\overline{Q}_1Q_2\overline{Q}_3D\,Z + P\,D$
$\hat{Z} = \overline{Q}_1Q_2\overline{Q}_3D\,Y + \overline{Q}_1Q_2\overline{Q}_3\overline{D}\,Z + Q_1Z + \overline{Q}_2Z + Q_3Z$
$\hat{G} = \overline{Q}_1G + \overline{Q}_3G + Q_1Q_2G + Q_1\overline{Q}_2Q_3\overline{D}\,L + Q_1\overline{Q}_2Q_3\overline{D}\,S$
$\hat{L} = \overline{Q}_1L + \overline{Q}_3L + \overline{Q}_2L + Q_1Q_2\overline{Q}_3\overline{D}\,G + Q_1Q_2Q_3Y$
$\hat{S} = \overline{Q}_1S + Q_2S + Q_3D\,S + \overline{Q}_3\overline{D}\,S + \overline{D}\,Y\,S + Q_1Q_2\overline{Q}_3D\,G + Q_1\overline{Q}_2\overline{Q}_3D\,Y$
$D_{set} = \overline{P}\,\overline{Q}_1\overline{Q}_2G\,Y + \overline{P}\,\overline{Q}_1Q_3G\,Y + P\,Q_3$
$D_{clear} = \overline{P}\,\overline{Q}_1\overline{Q}_2G\,\overline{Y} + \overline{P}\,\overline{Q}_1Q_3G\,\overline{Y} + P\,\overline{Q}_3$
$D_{clock} = P\,C + \overline{P}\,S$
$\hat{c}lock = clock$
$\hat{P} = P$

Table III. Microprogrammed Cell Equations for PLA Implementation

$\hat{X} = Q_1\overline{Q}_3Y(\overline{D}+Q_2) + D\ Q_3(\overline{Q}_1Q_2Q_3 + Q_1\overline{Q}_2 + Q_2X + \overline{Q}_1X\ Y) + \overline{Q}_2Q_3X(Q_1 + \overline{D}\ \overline{Y} + D) + \overline{Q}_1\overline{Q}_3(Q_2X + X\ Y + \overline{Q}_2\overline{Q}_3D\ Y)$
$\hat{Y} = \overline{P}\ \{D\ G\ L(\overline{Q}_1\overline{Q}_2\overline{X}\ \overline{Y} + \overline{Q}_1Q_2Q_3 + Q_1\overline{Q}_2\overline{Q}_3S) + \overline{D}\ X\ G(L\ Q_1\overline{Q}_2\overline{Q}_3 + L\ \overline{Q}_1\overline{Q}_2\overline{Y} + Q_1Q_2Q_3) + G\ \overline{Q}_2Q_3(Q_1D\ X + \overline{Q}_1Y\ L) + Y\ \overline{Q}_2(\overline{G} + \overline{Q}_1\overline{L} + \overline{Q}_3\overline{L} + \overline{Q}_1\overline{Q}_3X\ D) + Y\ Q_3(\overline{G} + \overline{Q}_1\overline{L} + Q_1\overline{Q}_2\overline{D} + Q_1Q_2) + \overline{Q}_3\overline{Q}_1(Y\ \overline{D}\ \overline{X} + Q_2D\ Z) + Q_1Q_2D\ Y\} + P\ D$
$\hat{Z} = \overline{Q}_1Q_2\overline{Q}_3(D\ Y + \overline{D}\ Z) + Z(Q_1 + \overline{Q}_2 + Q_3) = \overline{Q}_1\overline{Q}_3(D\ Y\ Q_2 + \overline{D}\ Z\ Q_2) + Z(Q_1 + \overline{Q}_2 + Q_3)$
$\hat{G} = G(\overline{Q}_1 + \overline{Q}_3 + Q_1Q_2) + \overline{D}\ Q_1\overline{Q}_2(Q_3L + Q_3S)$
$\hat{L} = L(\overline{Q}_1 + \overline{Q}_3 + \overline{Q}_2) + Q_1Q_2(\overline{Q}_3\overline{D}\ G + Q_3Y)$
$\hat{S} = S(\overline{Q}_1 + Q_2) \pm S\ Q_3D + S\ \overline{D}(\overline{Q}_3 + Y) + Q_1\overline{Q}_3(D\ Q_2G + \overline{Q}_2\overline{Q}_3D\ Y)$
$D_{set} = \overline{P}\ \overline{Q}_1G\ Y(\overline{Q}_2 + Q_3) + P\ Q_3 = Y(\overline{P}\ \overline{Q}_1G\ Q_2 + \overline{P}\ \overline{Q}_1G\ Q_3) + P\ Q_3$
$D_{clear} = \overline{P}\ \overline{Q}_1G\ \overline{Y}(\overline{Q}_2 + Q_3) + P\ \overline{Q}_3 = \overline{Y}(\overline{P}\ \overline{Q}_1G\ Q_2 + \overline{P}\ \overline{Q}_1G\ Q_3) + P\ \overline{Q}_3$
$D_{clock} = P\ C + \overline{P}\ S$

Table IV. Microprogrammed Cell Equations for Random Logic Implementation

$\hat{X} = \overline{S}_5\overline{S}_6\overline{S}_7I_1 + \overline{S}_5\overline{S}_6S_7I_2 + \overline{S}_5S_6\overline{S}_7I_3 + \overline{S}_5S_6S_7I_4 + S_5I_5$
$\hat{Y} = \overline{S}_8\overline{S}_9\overline{S}_{10}I_1 + \overline{S}_8\overline{S}_9S_{10}I_2 + \overline{S}_8S_9\overline{S}_{10}I_3 + \overline{S}_8S_9S_{10}I_4 + S_8I_5$
$\hat{Z} = \overline{S}_1\overline{S}_2\overline{S}_3\overline{S}_4 + S_1\overline{S}_3\overline{S}_4\overline{Y} + \overline{S}_1\overline{S}_2\overline{S}_4X + \overline{S}_1\overline{S}_2\overline{S}_4Y + S_1S_2\overline{S}_3\overline{S}_4\overline{X} + \overline{S}_3\overline{S}_4\overline{X}\,\overline{Y} + S_1\overline{S}_4X\,\overline{Y} + S_1S_2\overline{S}_4\overline{X}\,Y + S_4Q$
$\hat{Q} = S_4X\,\overline{Y} + Q\,\overline{S}_4 + Q\,\overline{Y}$

Table V. Cobweb Cell Equations for PLA Implementation

FUNCTION	α	β	γ	OPERATION PERFORMED
0	0	0	0	NO OP
1	0	0	1	MULTIPLY
2	0	1	0	LOAD MULTIPLIER, CLEAR HIGH PRODUCT, CLEAR ADDER
4	1	0	0	OUTPUT HIGH PRODUCT, SHIFT LOW PRODUCT TO HIGH
6	1	1	0	LOAD MULTIPLIER, SHIFT PREVIOUS CONTENTS TO MULTIPLICAND

TABLE VI. SEVEN-BIT MULTIPLIER CONTROL FUNCTIONS

CELL TYPE	DIMENSIONS						AREA/ $\sqrt{AREA}$			
							$100m\ell^2$/bit of storage		$300m\ell^2$/bit of storage	
	INPUTS	OUTPUTS	ROWS	PRODUCT TERMS	LOGIC LEVELS	# BITS STORAGE	$1m\ell^2$/bit	$2m\ell^2$/bit	$1m\ell^2$/bit	$2m\ell^2$/bit
MICRO-PROGRAMMED	12	9	33	63	2	4	2479/50	4558/68	3279/57	5358/73
COBWEB	18	4	40	23	4	10	1920/44	2840/53	3920/63	4840/70
PROGRAMMABLE	9	7	25	15	2	2	575/24	950/31	975/31	1350/37

TABLE VII. AREA CALCULATIONS FOR PLA IMPLEMENTATION

CELL TYPE	NUMBER OF GATES					LOGIC LEVELS	TOTAL # PINS	# BITS STORAGE	AREA/ $\sqrt{AREA}$			
									$100 m\ell^2$/bit of storage		$300 m\ell^2$/bit of storage	
	1 INPUT	2 INPUT	3 INPUT	4 INPUT	6 INPUT				50% Wiring	80% Wiring	50% Wiring	80% wiring
MICRO-PROGRAMMED	0	43	26	20	1	6	340	4	10192/101	24880/158	10992/105	25680/160
COBWEB	0	34	3	1	0	7	129	14	5115/71.5	10688/103	7915/89	13488/116
PROGRAMMABLE	5	8	7	4	0	2	82	2	2361/49	5904/77	2361/49	5904/77

TABLE VIII. AREA CALCULATIONS FOR RANDOM LOGIC IMPLEMENTATION

ROW	TASK	CELLULAR ARRAY	NUMBER OF CELLS	ARRAY SIZE-PLA IMPLEMENTATION (UNITS ARE INCHES^2)			
				100 Mils^2/bit in Shift Register		300 Mils^2/bit in Shift Register	
				1 Mils^2/bit in PLA array	2 Mils^2/bit in PLA array	1 Mils^2/bit in PLA array	2 Mils^2/bit in PLA array
1	SEVEN BIT MULTIPLIER	MICRO-PROGRAMMED	138	0.342	0.629	0.453	0.739
2		COBWEB	432	0.829	1.071	1.693	2.090
3		PROGRAMMABLE	342	0.199	0.327	0.341	0.469
4	31-BIT SEQUENCE DETECTOR	MICRO-PROGRAMMED	165	0.409	0.752	0.541	0.884
5		COBWEB	480	0.922	1.363	1.882	2.323
6		PROGRAMMABLE	610	0.356	0.585	0.613	0.843
7	31-BIT FEEDBACK SHIFT REGISTER DECODER	MICRO-PROGRAMMED	720	1.784	3.282	2.361	3.858
8		COBWEB	406	0.780	1.153	1.592	1.965
9		PROGRAMMABLE	512	0.297	0.489	0.508	0.700

TABLE IX. SUMMARY OF PLA IMPLEMENTATIONS

ROW	TASK	CELLULAR ARRAY	NUMBER OF CELLS	ARRAY SIZE-RANDOM LOGIC (UNITS ARE INCHES^2) 100 Mils^2/bit in Shift Register 50% Wiring	100 Mils^2/bit in Shift Register 80% Wiring	300 Mils^2/bit in Shift Register 50% Wiring	300 Mils^2/bit in Shift Register 80%Wiring
1	SEVEN BIT MULTIPLIER	MICRO-PROGRAMMED	138	1.406	3.517	1.517	3.544
2		COBWEB	432	2.210	4.617	3.419	5.827
3		PROGRAMMABLE	342	0.809	1.824	0.813	1.829
4	31-BIT SEQUENCE DETECTOR	MICRO-PROGRAMMED	165	1.682	4.105	1.814	4.237
5		COBWEB	480	2.455	5.130	3.799	6.474
6		PROGRAMMABLE	610	1.446	3.256	1.459	3.269
7	31-BIT FEEDBACK SHIFT REGISTER DECODER	MICRO-PROGRAMMED	720	7.338	17.91	7.914	18.48
8		COBWEB	406	2.077	4.339	3.213	5.476
9		PROGRAMMABLE	512	1.094	2.731	1.100	2.737

TABLE X. SUMMARY OF RANDOM LOGIC IMPLEMENTATIONS

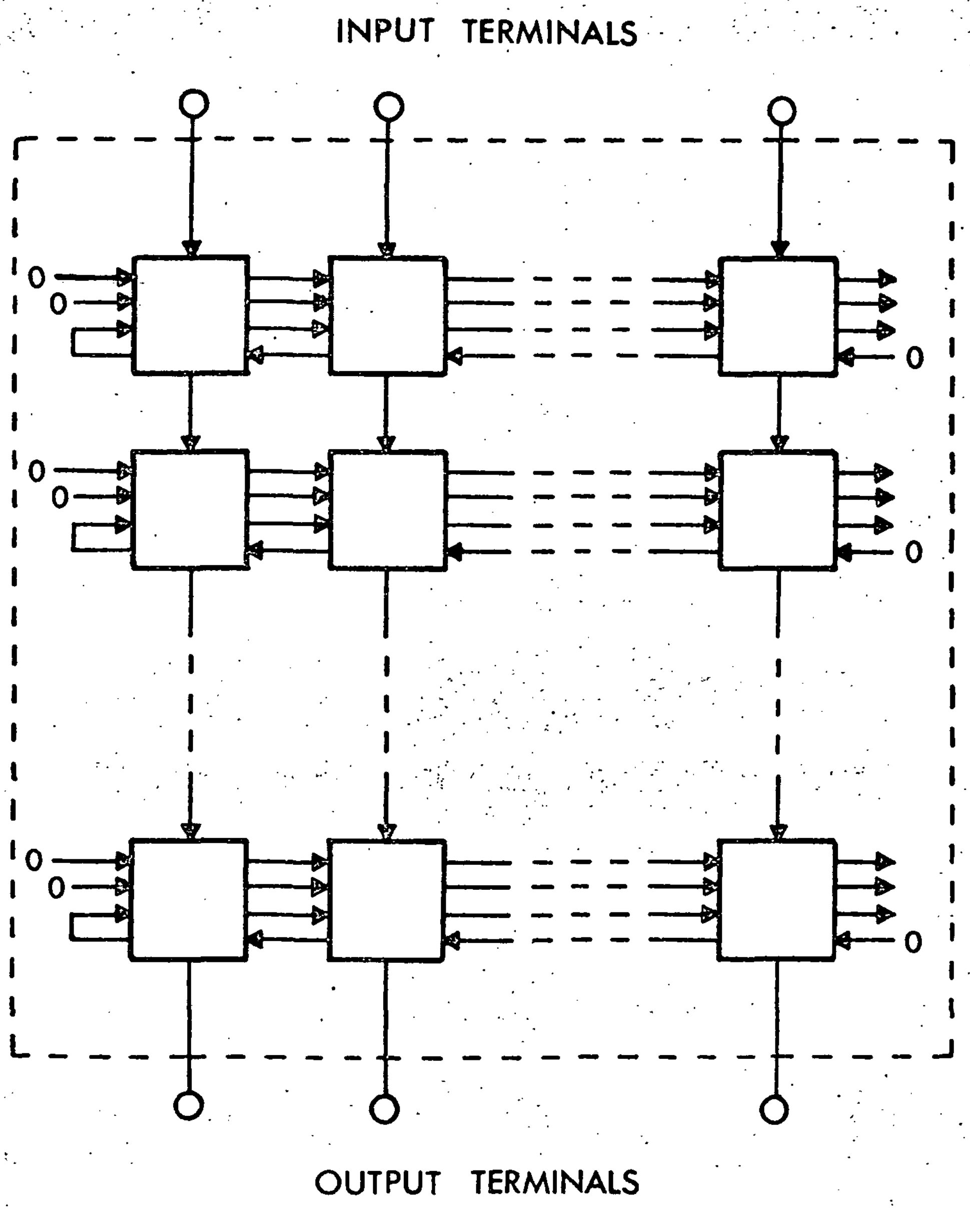


Fig. 1. Microprogrammed array interconnection structure.

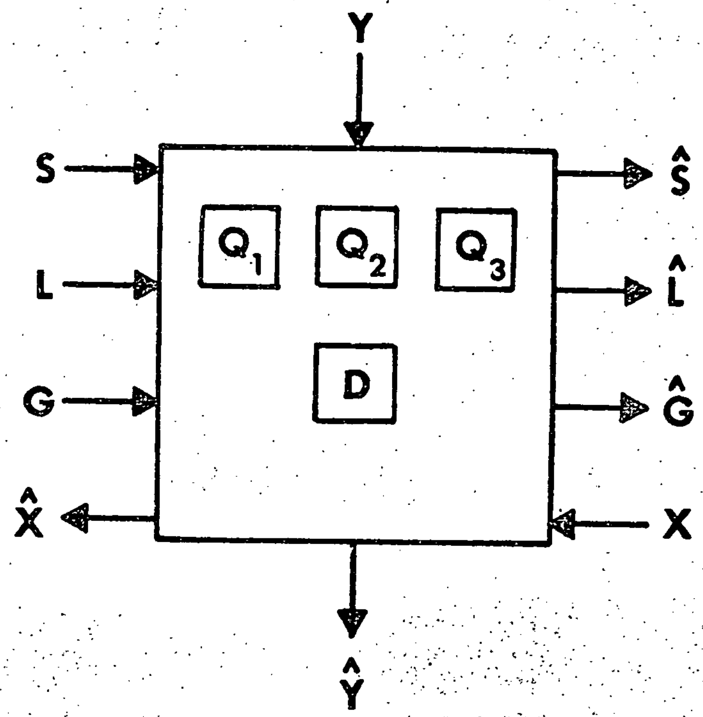


Fig. 2. Microprogrammed cell variables.

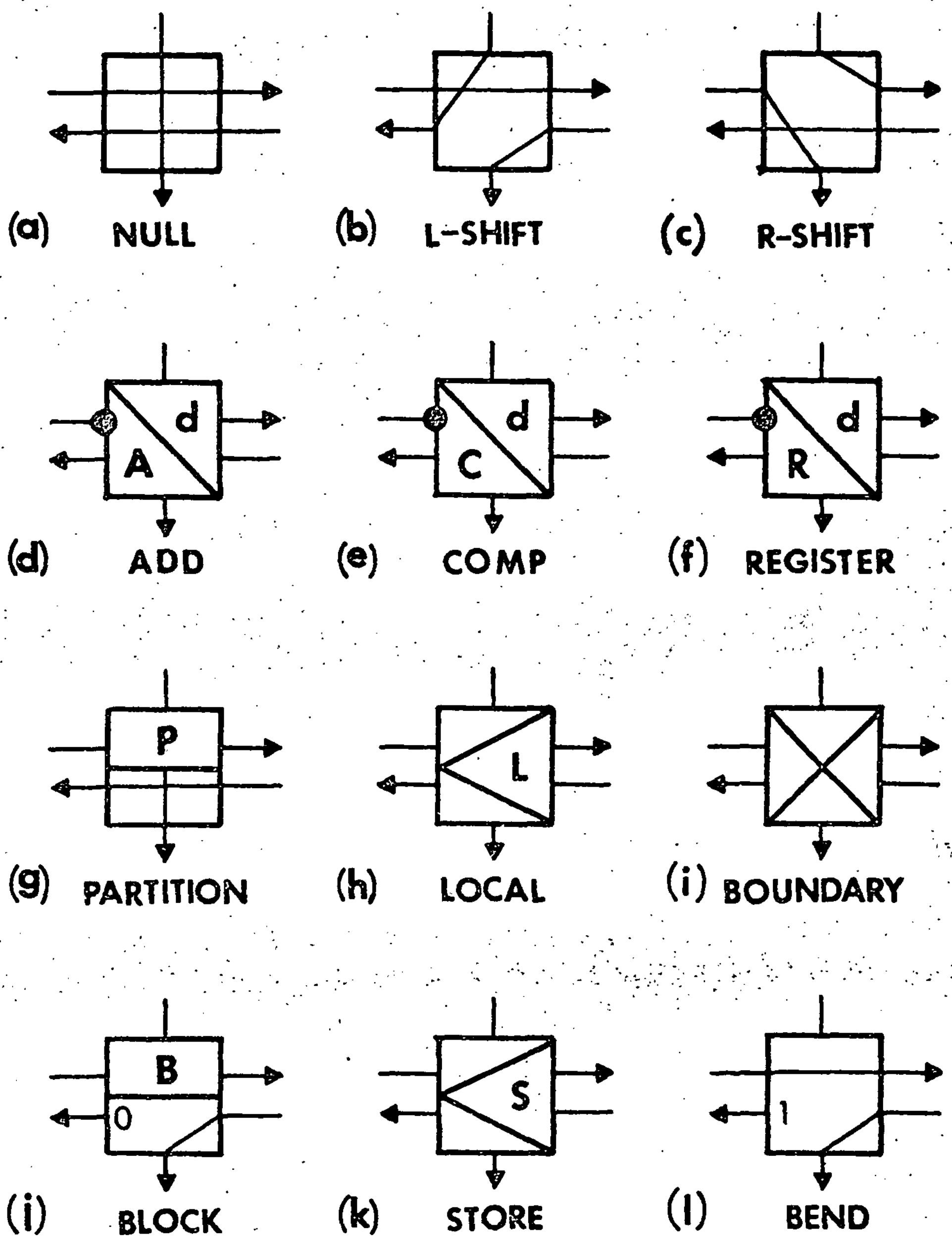


Fig. 3. Microprogrammed active cell symbols.

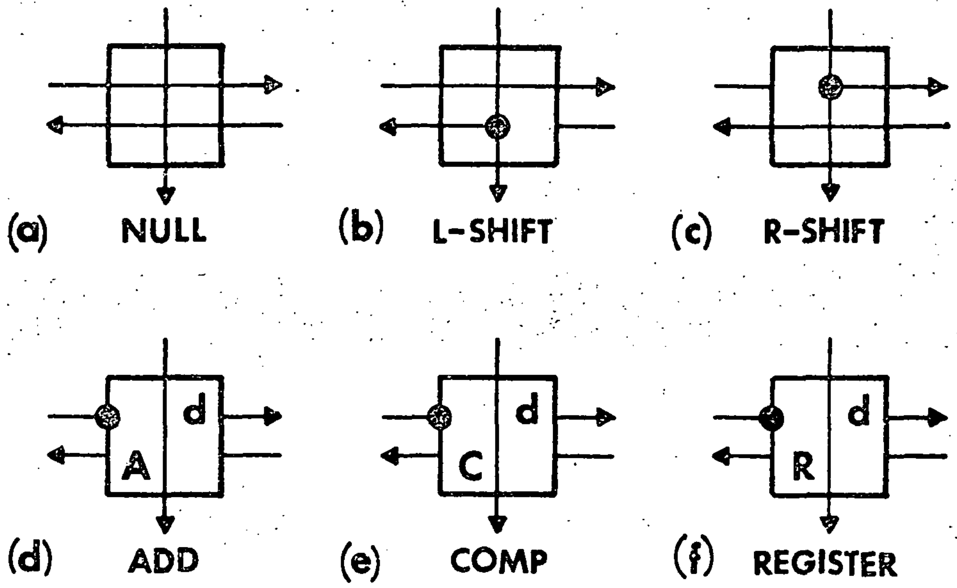


Fig. 4. Microprogrammed inactive cell symbols.

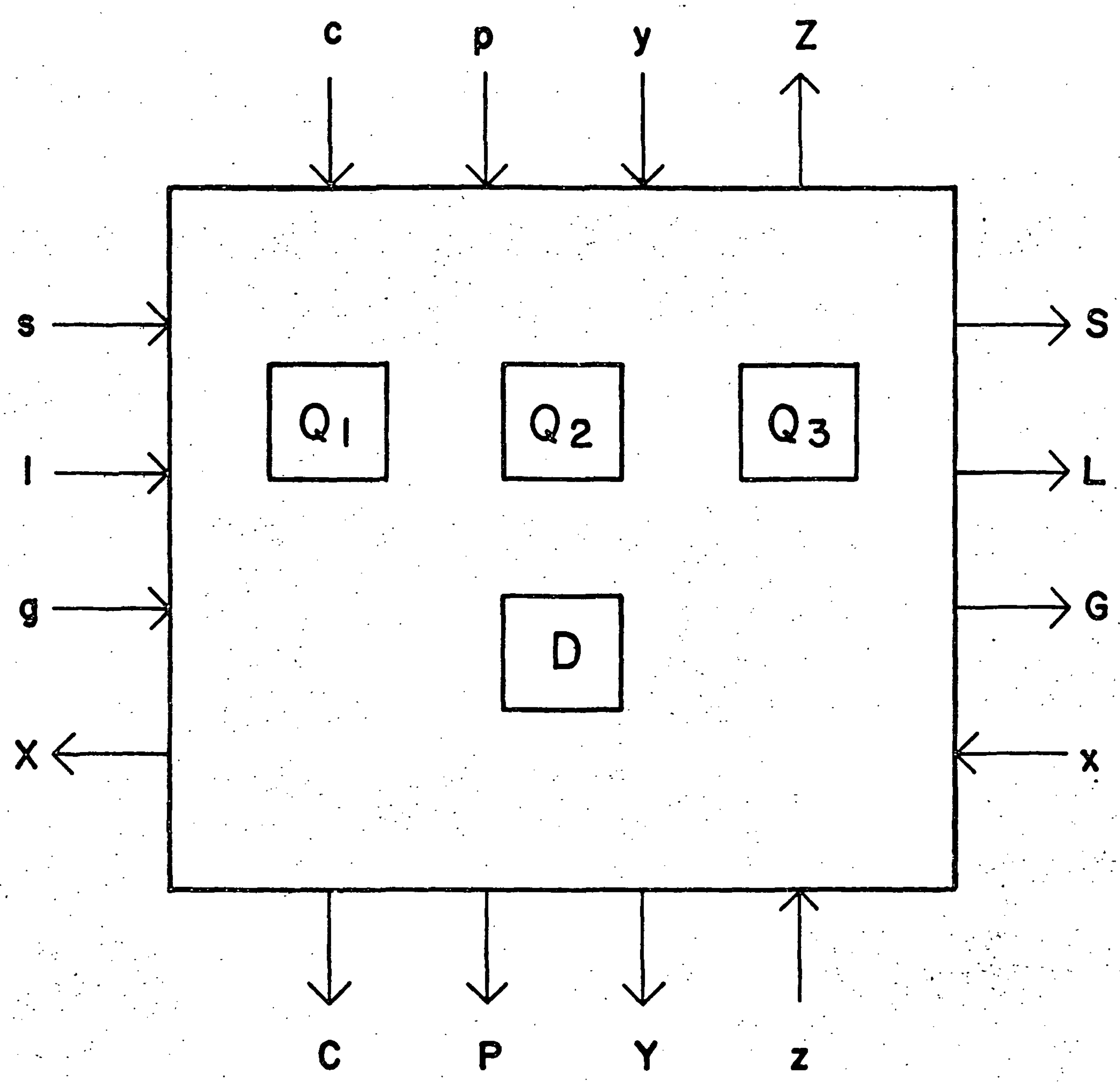


Fig. 5. Modified microprogrammed cell variables.

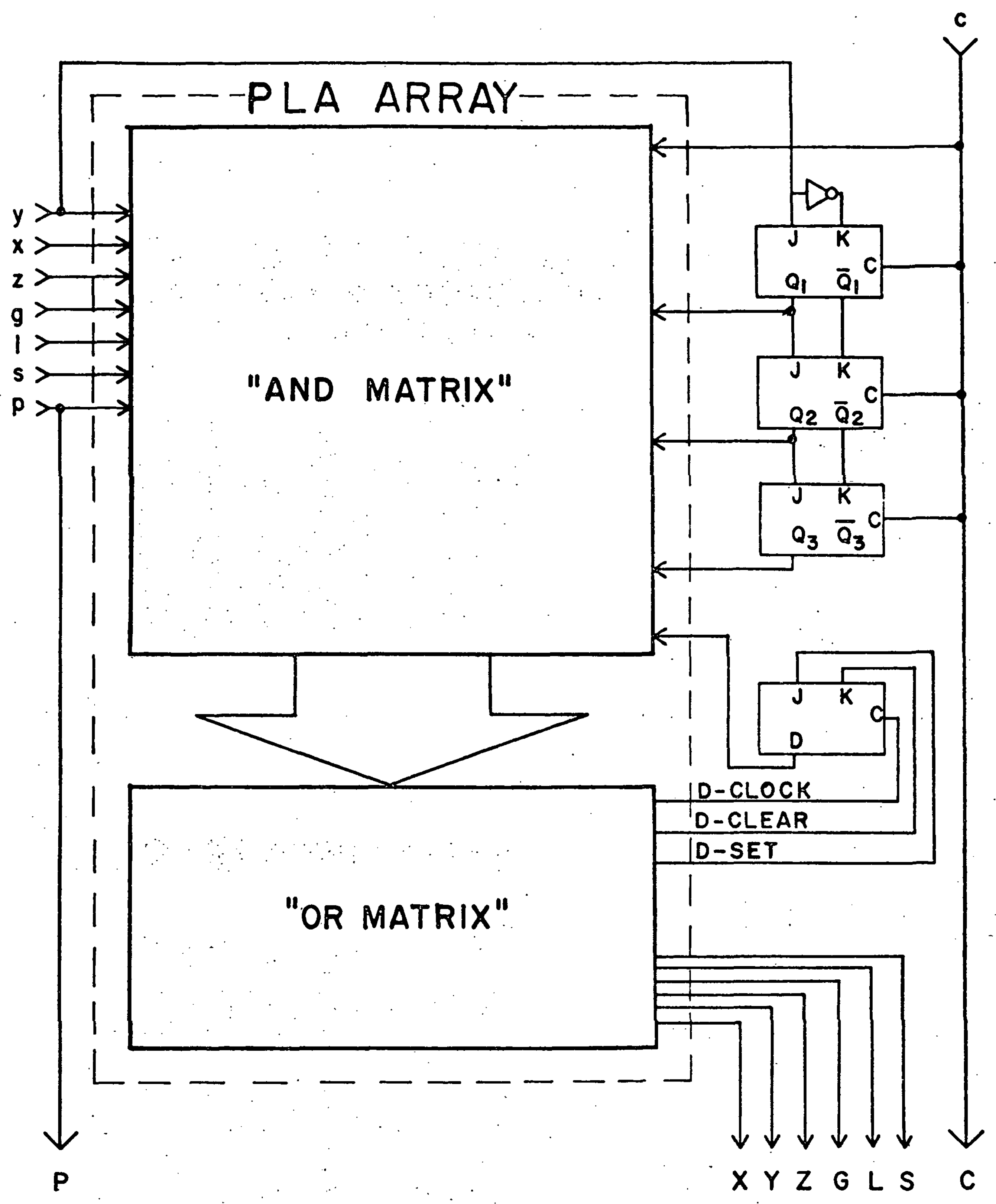


Fig. 6. PLA realization for the microprogrammed cell.

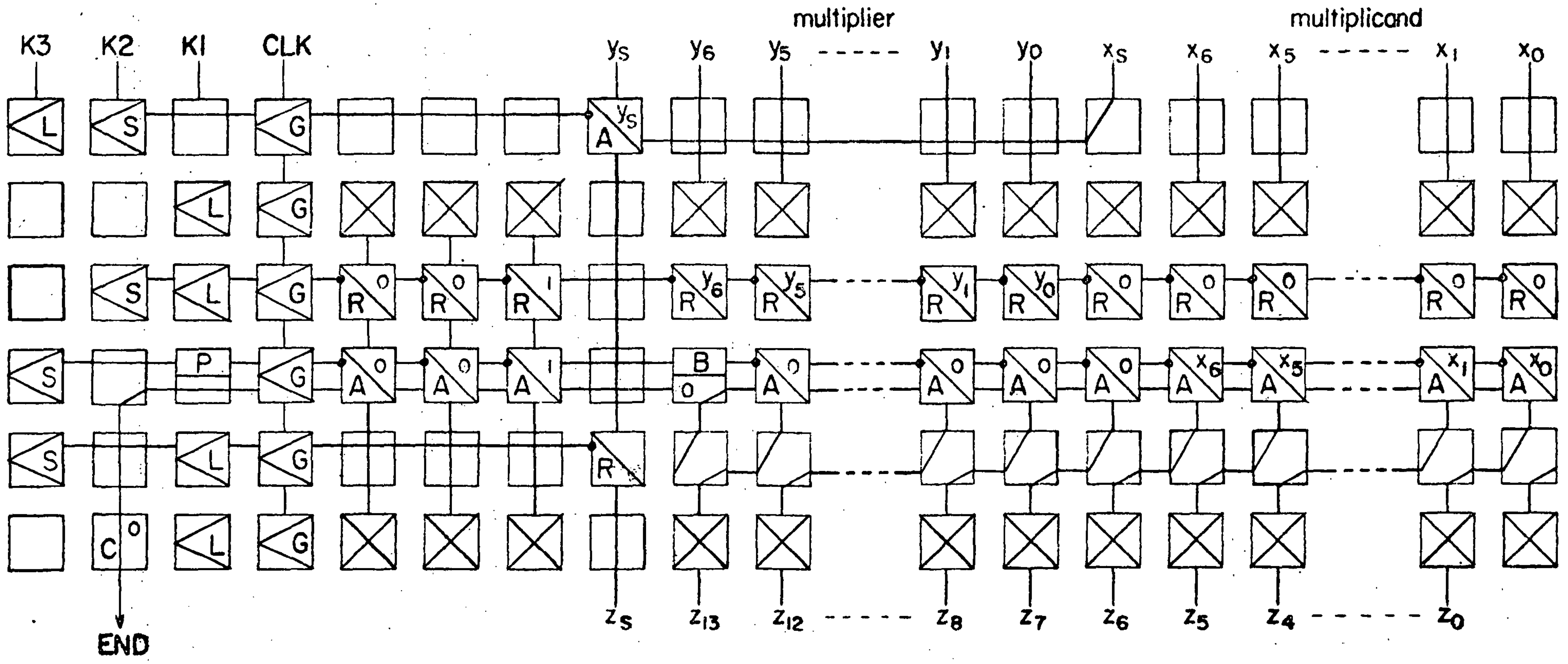


FIG. 7. MICROPROGRAMMED ARRAY SERIAL MULTIPLIER USING A PARALLEL ADDER.

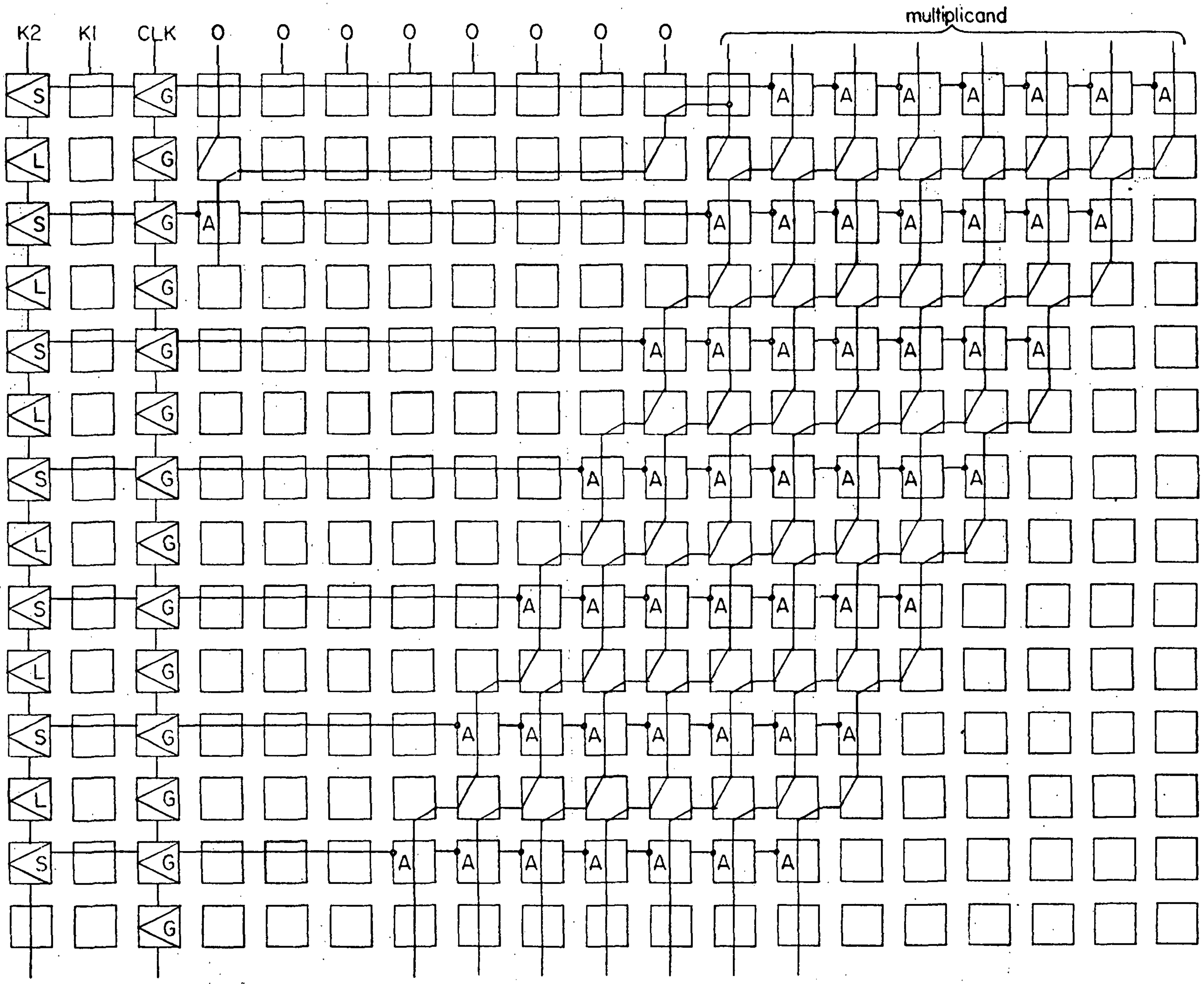


FIG. 8. MICROPROGRAMMED ARRAY PARALLEL MULTIPLIER
PHASE 1 - LOAD MULTIPLICAND (K2,K1)=(L,0).

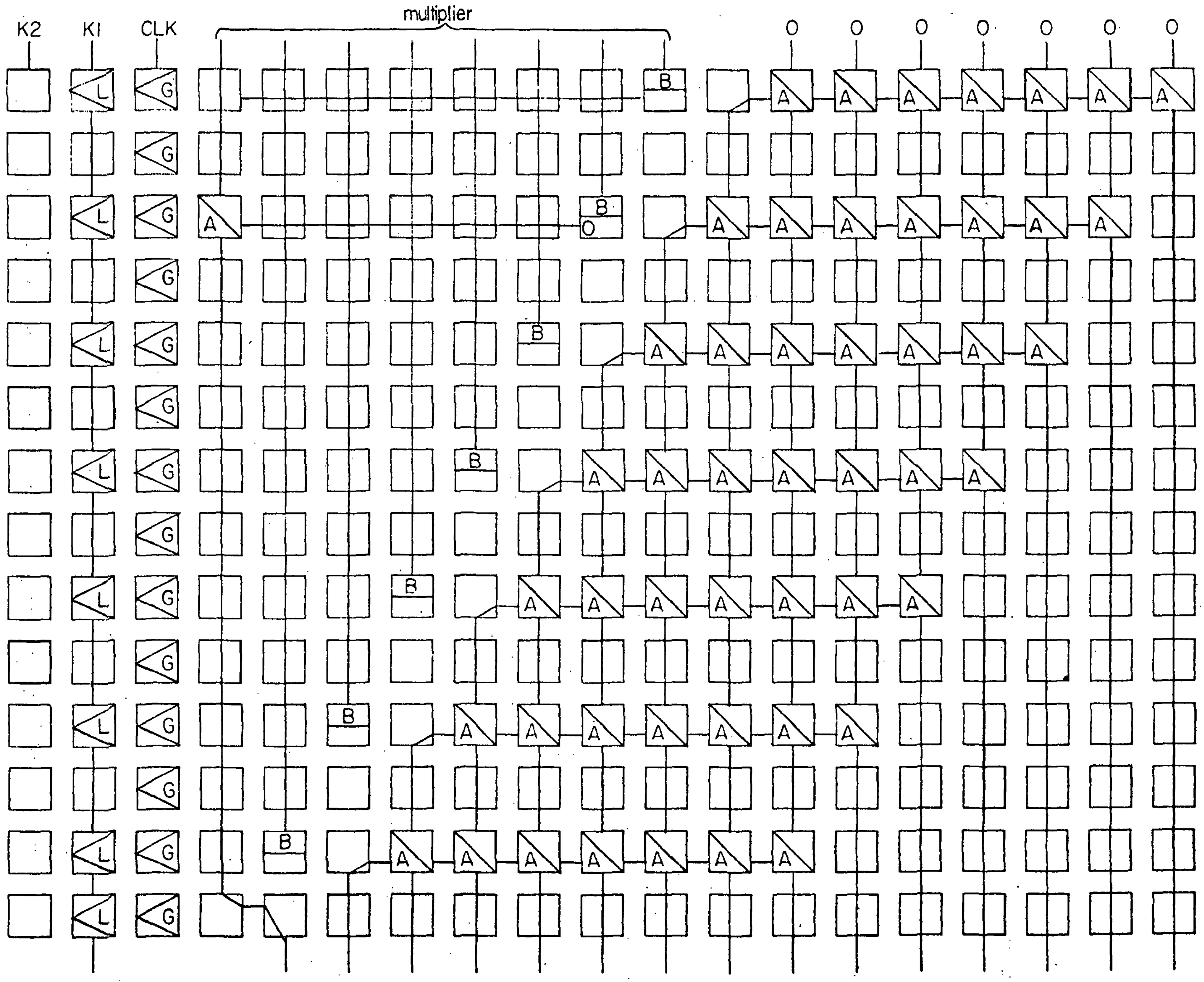


Fig. 9. Microprogrammed array parallel multiplier:
phase 2 - execute (K2,K1)=(0,1).

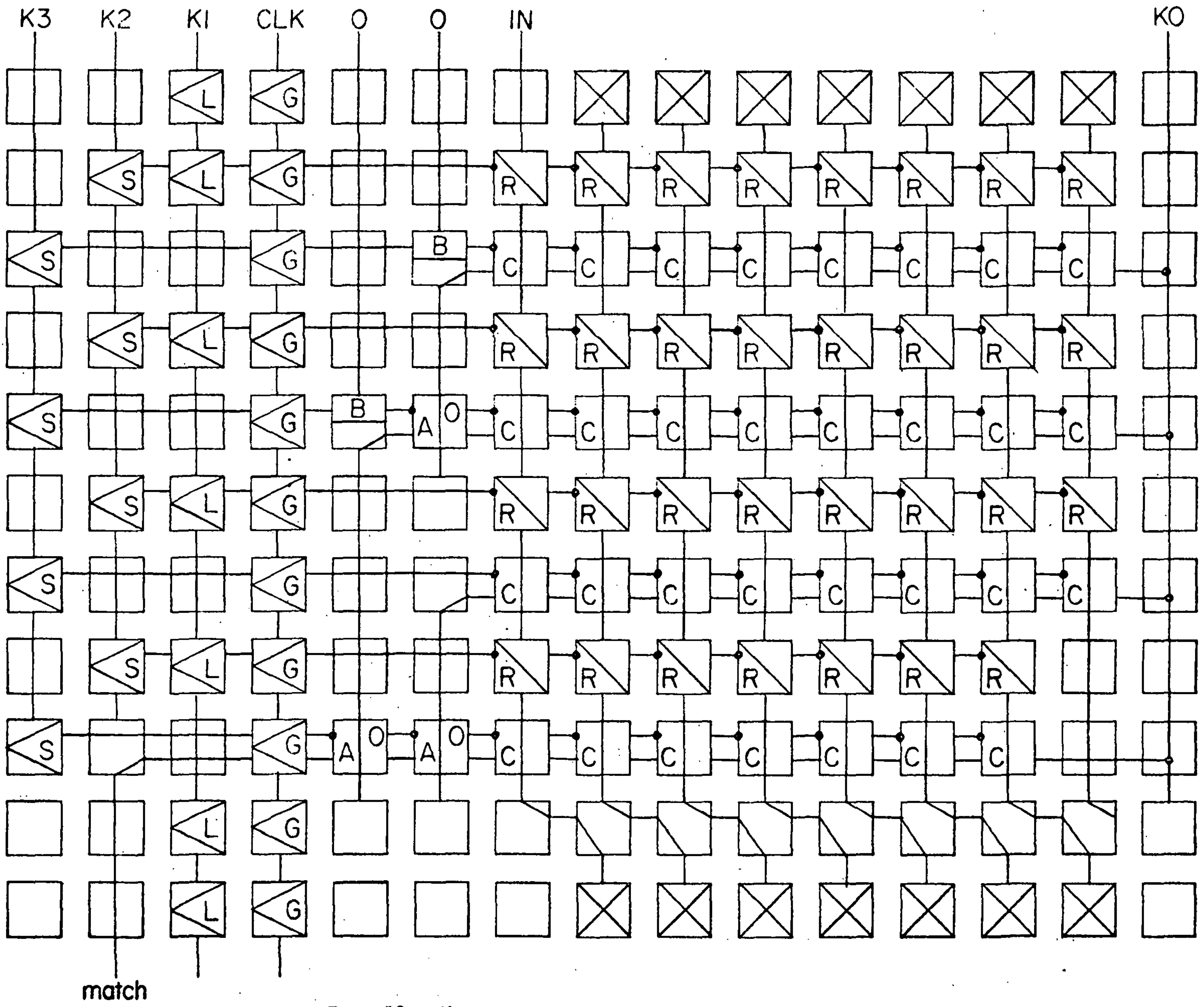


FIG. 10. MICROPROGRAMMED ARRAY - SEQUENCE DETECTOR:
PHASE 1 - SHIFT $(K_3K_2K_1K_0)=(0111)$.
PHASE 2 - SET SEQUENCE (1010).

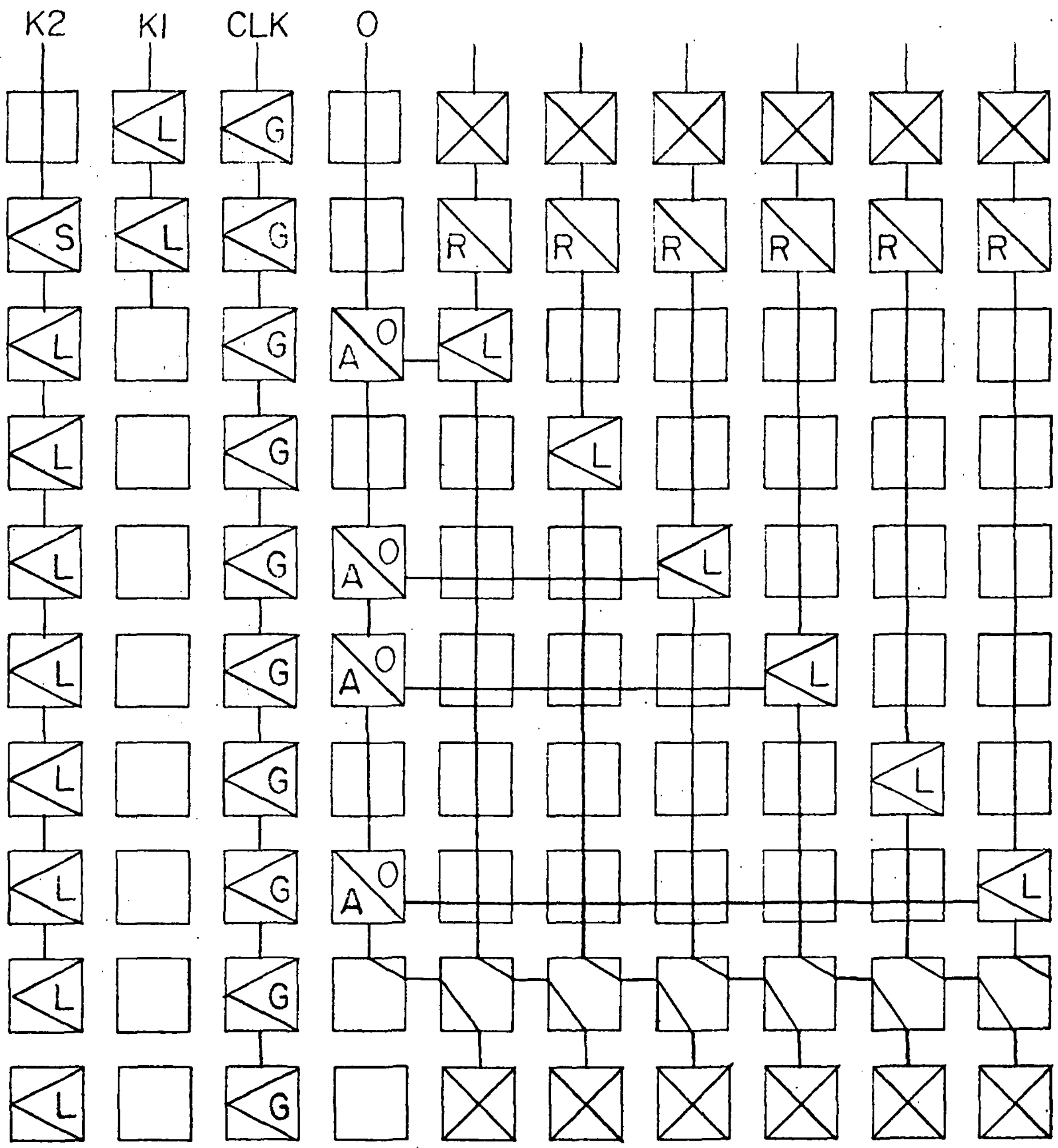


Fig. 11. Microprogrammed array: linear FSR (load - 10, read - 01, ex - 11)(parallel initial load).

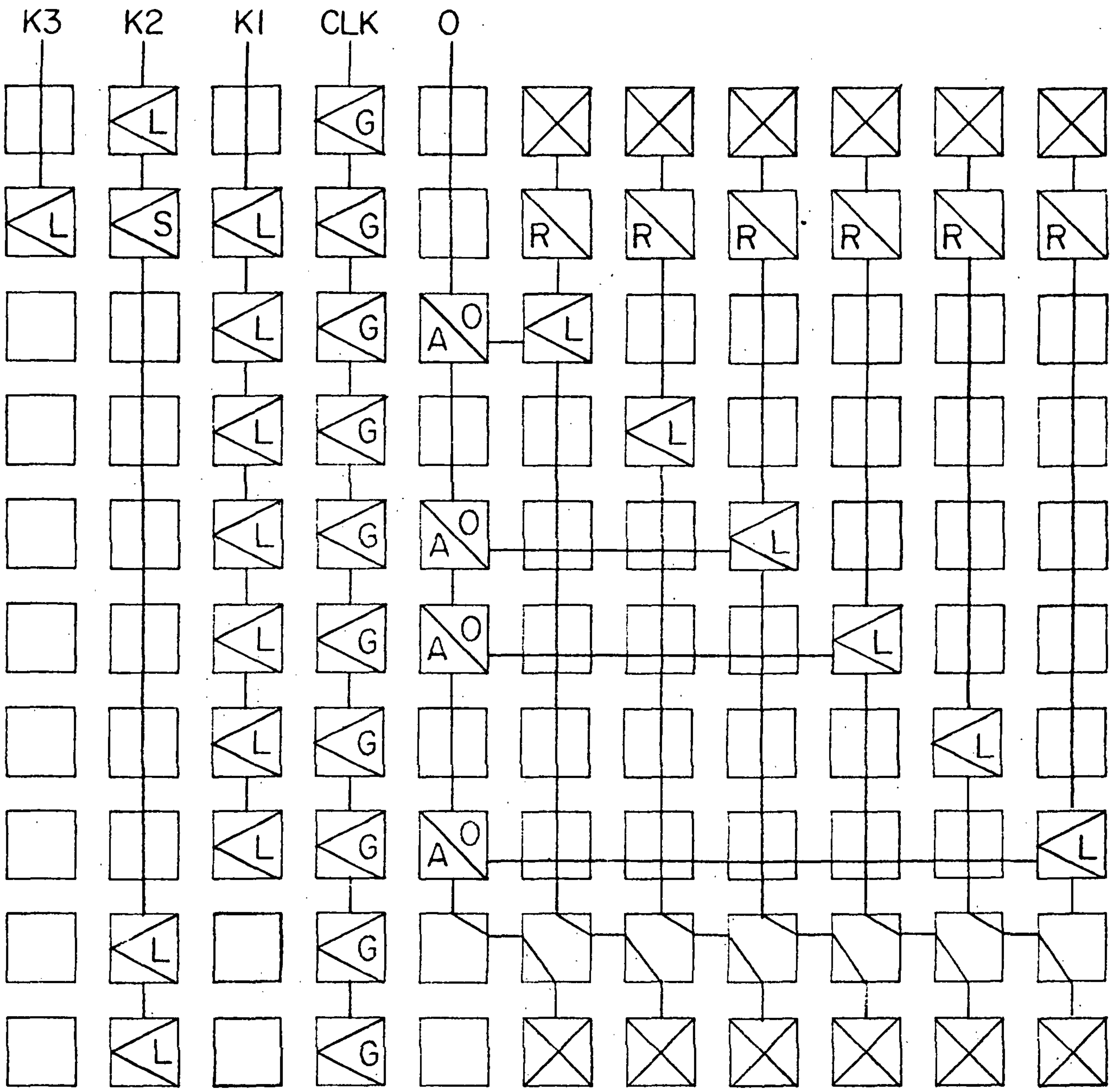


Fig. 12. Microprogrammed array: linear FSR

(load - 110, read - 100, ex - 011)(serial load).

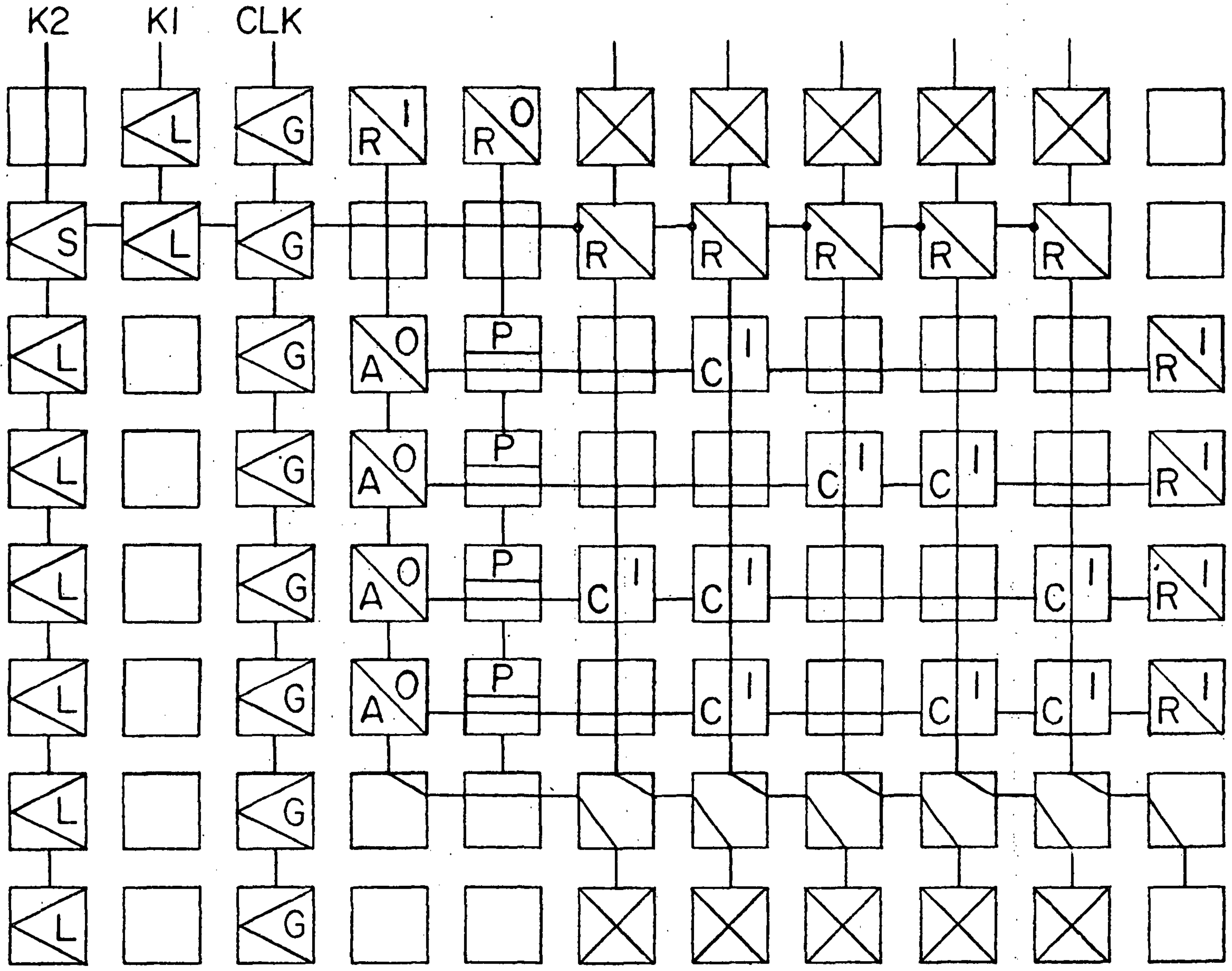


Fig. 13. Microprogrammed array: nonlinear FSR
(load - 10, read - 01, ex - 11)(parallel initial load).

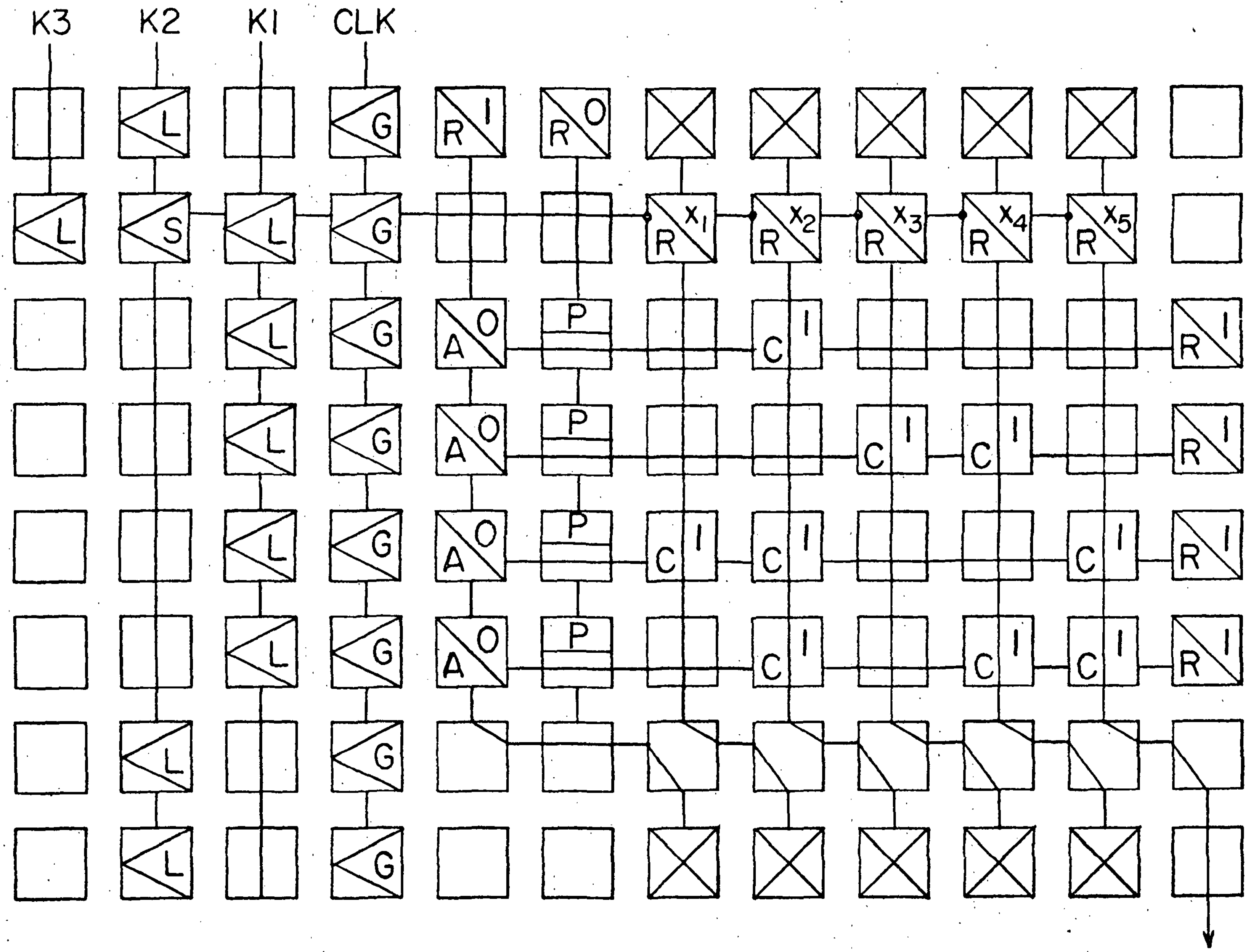


Fig. 14. Microprogrammed array: nonlinear FSR
(load - 110, read - 100, ex - 011)(serial load).

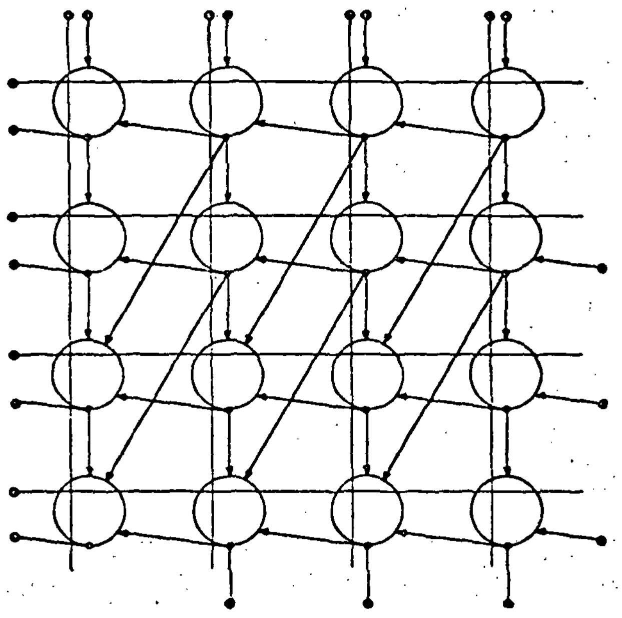


Fig. 15. Structure of the cobweb array.

INDEX	S_1	S_2	S_3	S_4	z
0	0	0	0	0	1
1	0	0	0	1	y'
2	0	0	1	0	$x'+y'$
3	0	0	1	1	$x'y'$
4	0	1	0	0	$x+y$
5	0	1	0	1	xy'
6	0	1	1	0	$z \oplus y$
7	0	1	1	1	0
F	1	1	0	1	$x=S$, $y=R$

Fig. 16. Functions of one cobweb cell.

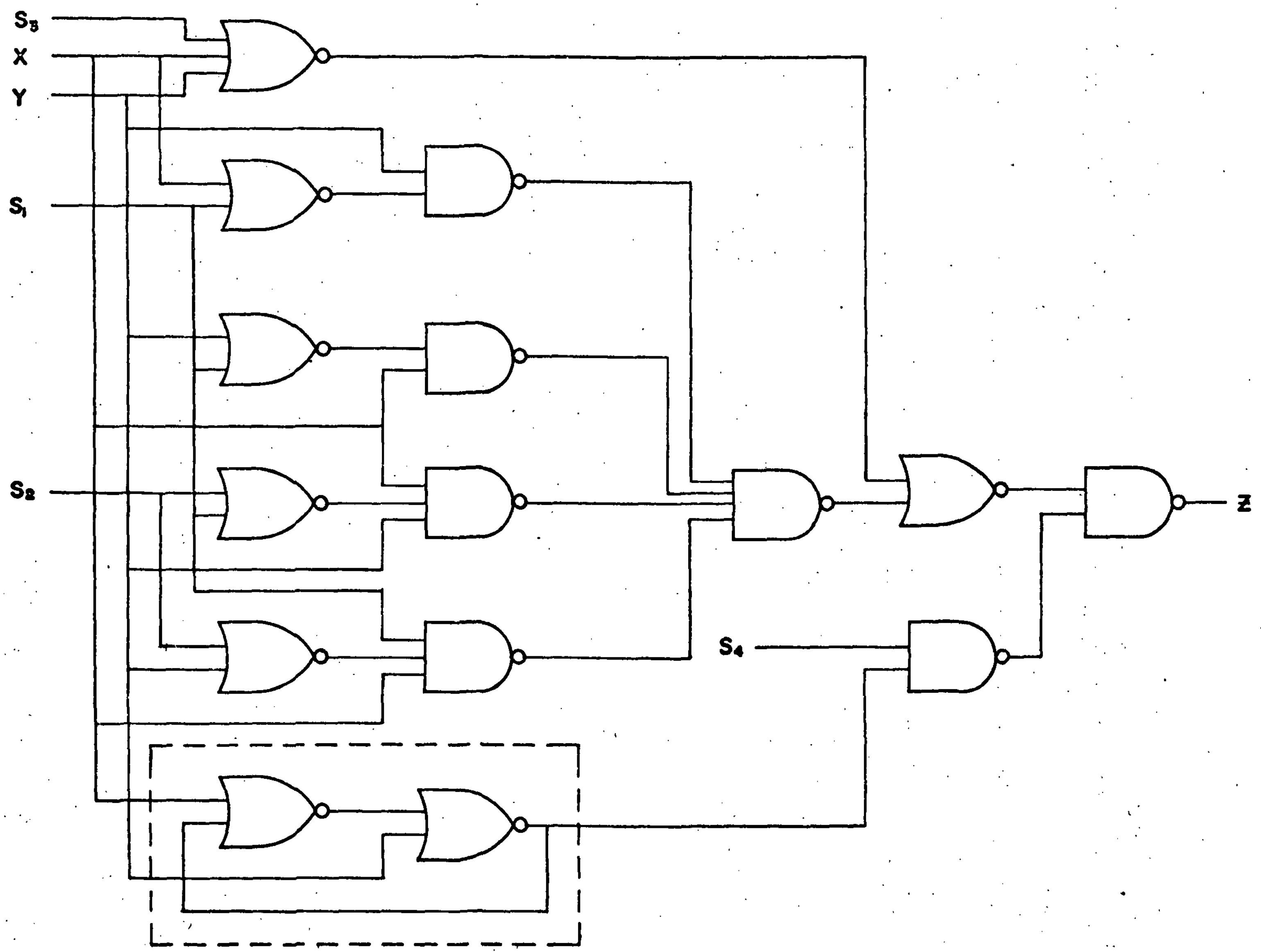


FIG. 17. NAND-NOR REALIZATION FOR THE INTERNAL PORTION OF ONE COBWEB CELL.

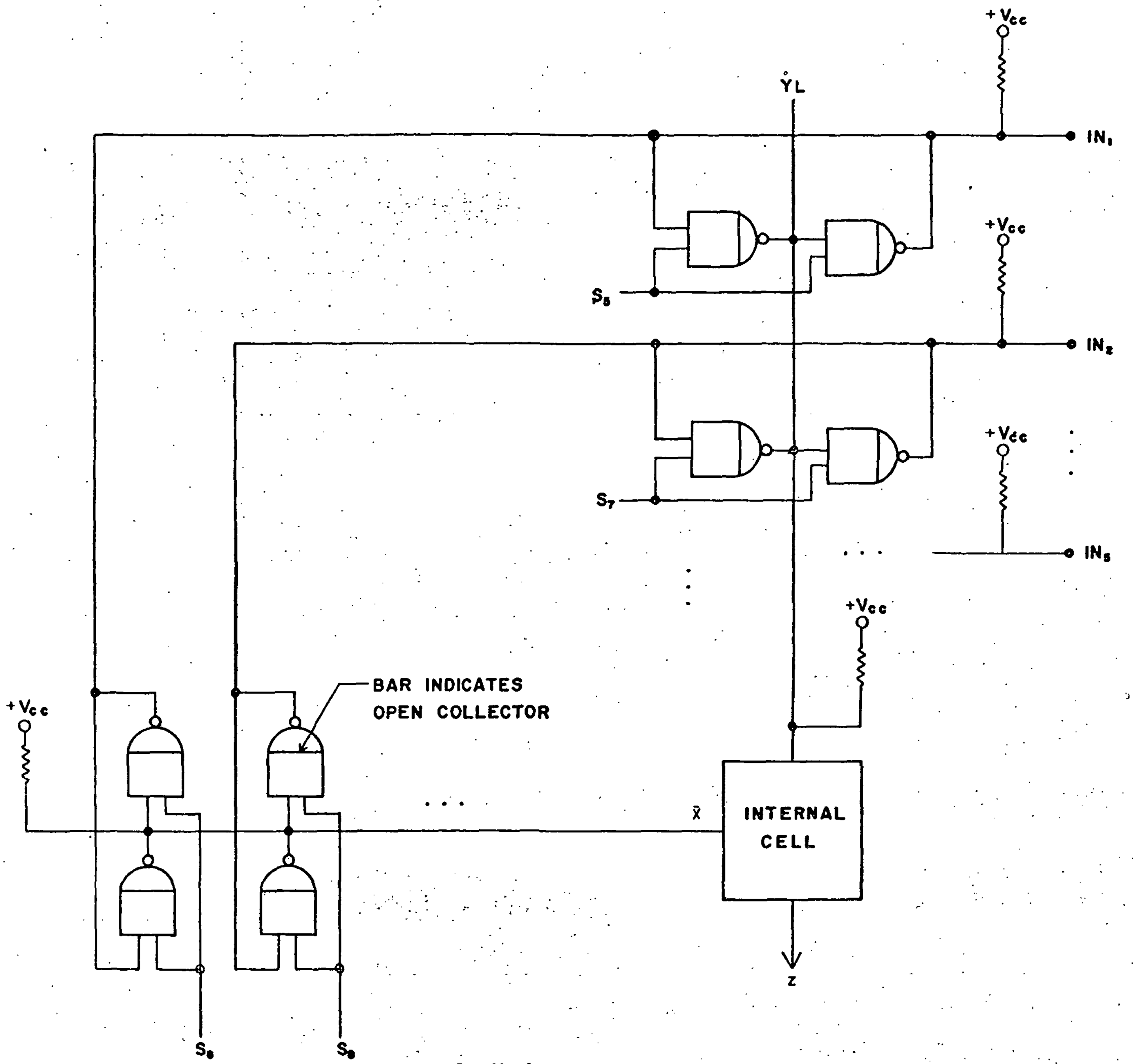
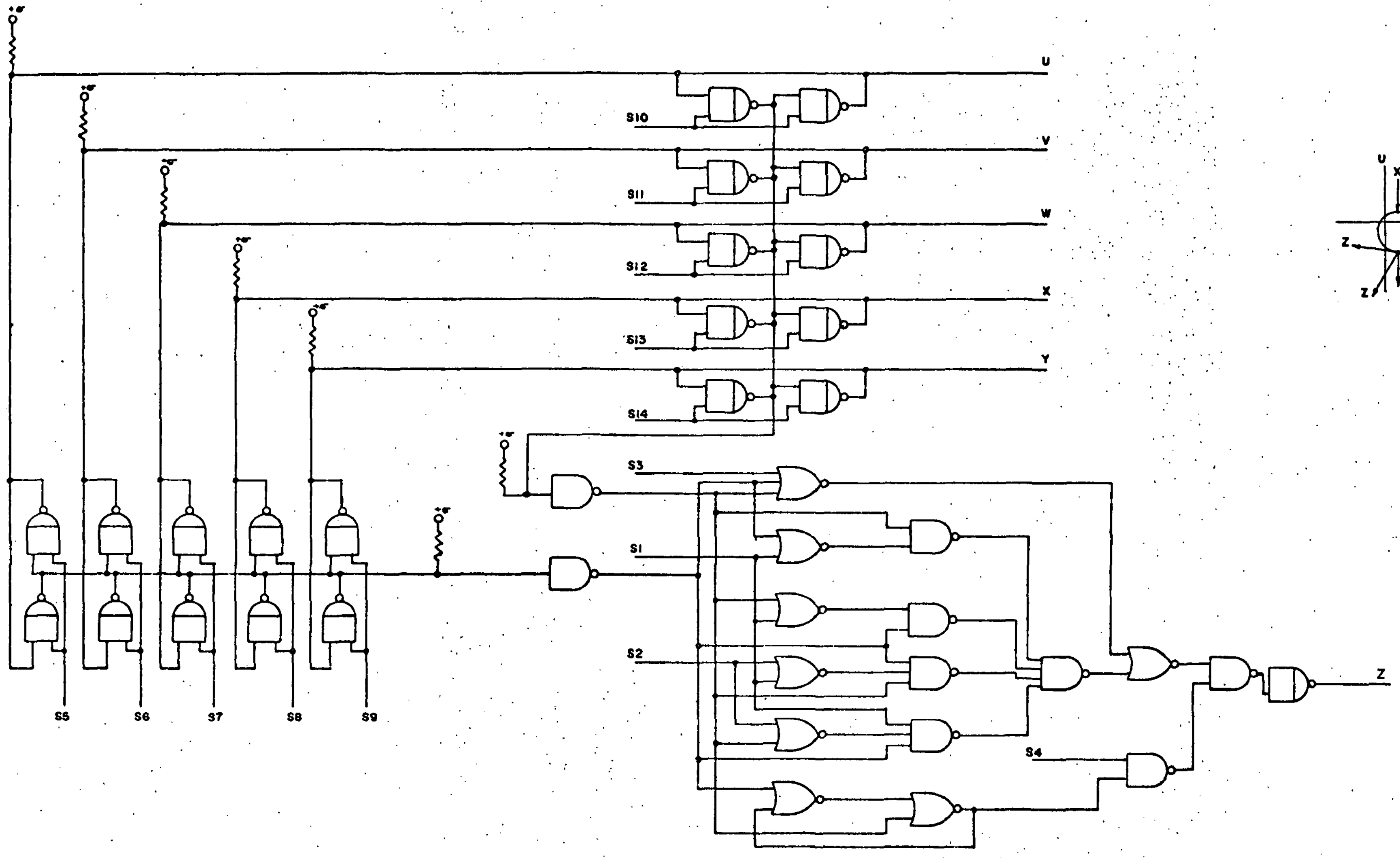
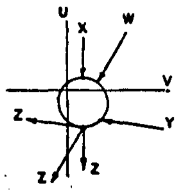


Fig. 18. Input selection logic for the cobweb cell.





Bar indicates open collector output

FIG 19. TTL REALIZATION FOR THE COBWEB CELL USING AN R-S FLIP-FLOP.

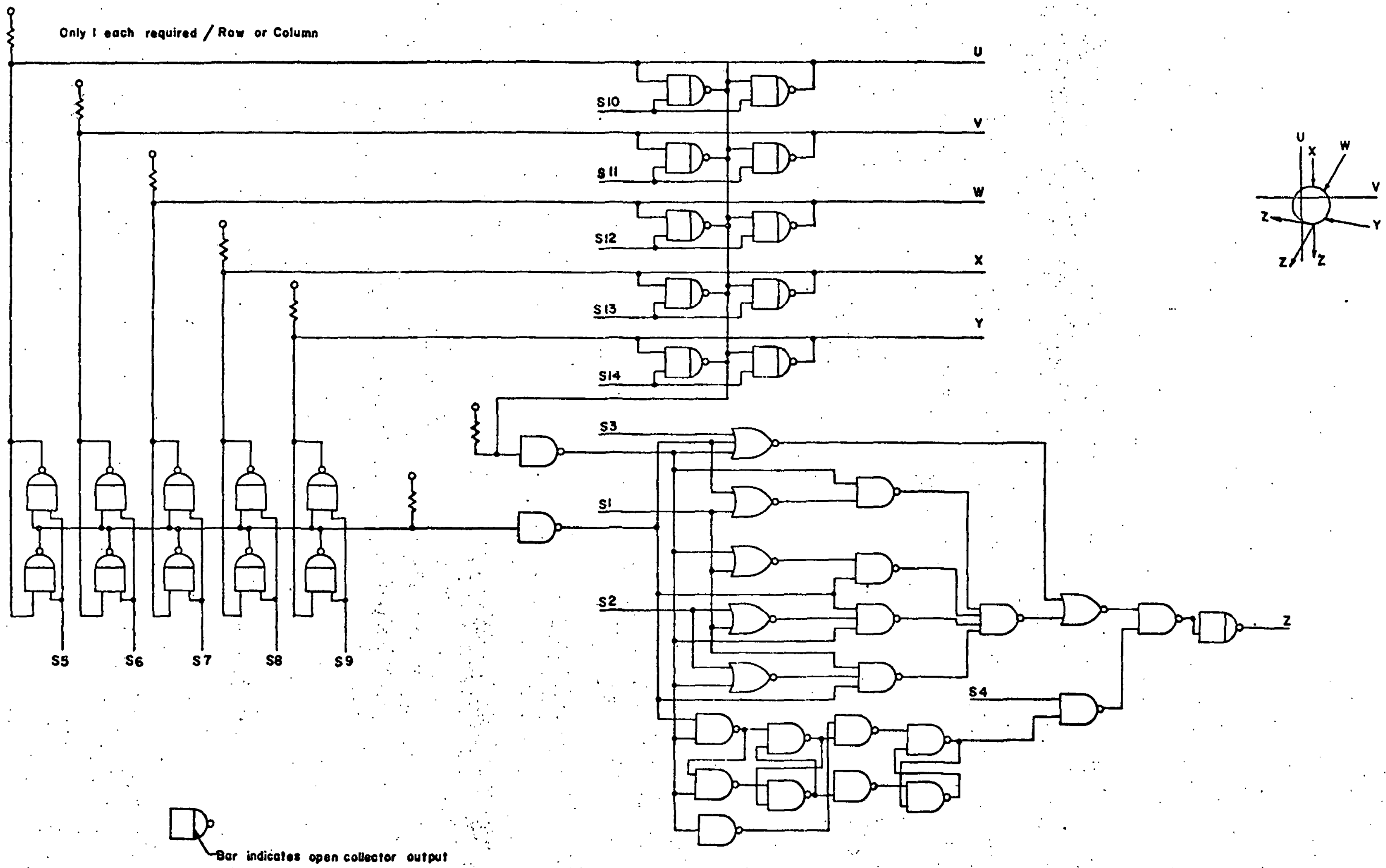

Only 1 each required / Row or Column
U
V
W
X
Y
S10
S11
S12
S13
S14
S3
S1
S2
S4
S5
S6
S7
S8
S9
Z
Bar indicates open collector output

FIG. 20. TTL REALIZATION FOR THE COBWEB CELL USING A MASTER-SLAVE D FLIP-FLOP.

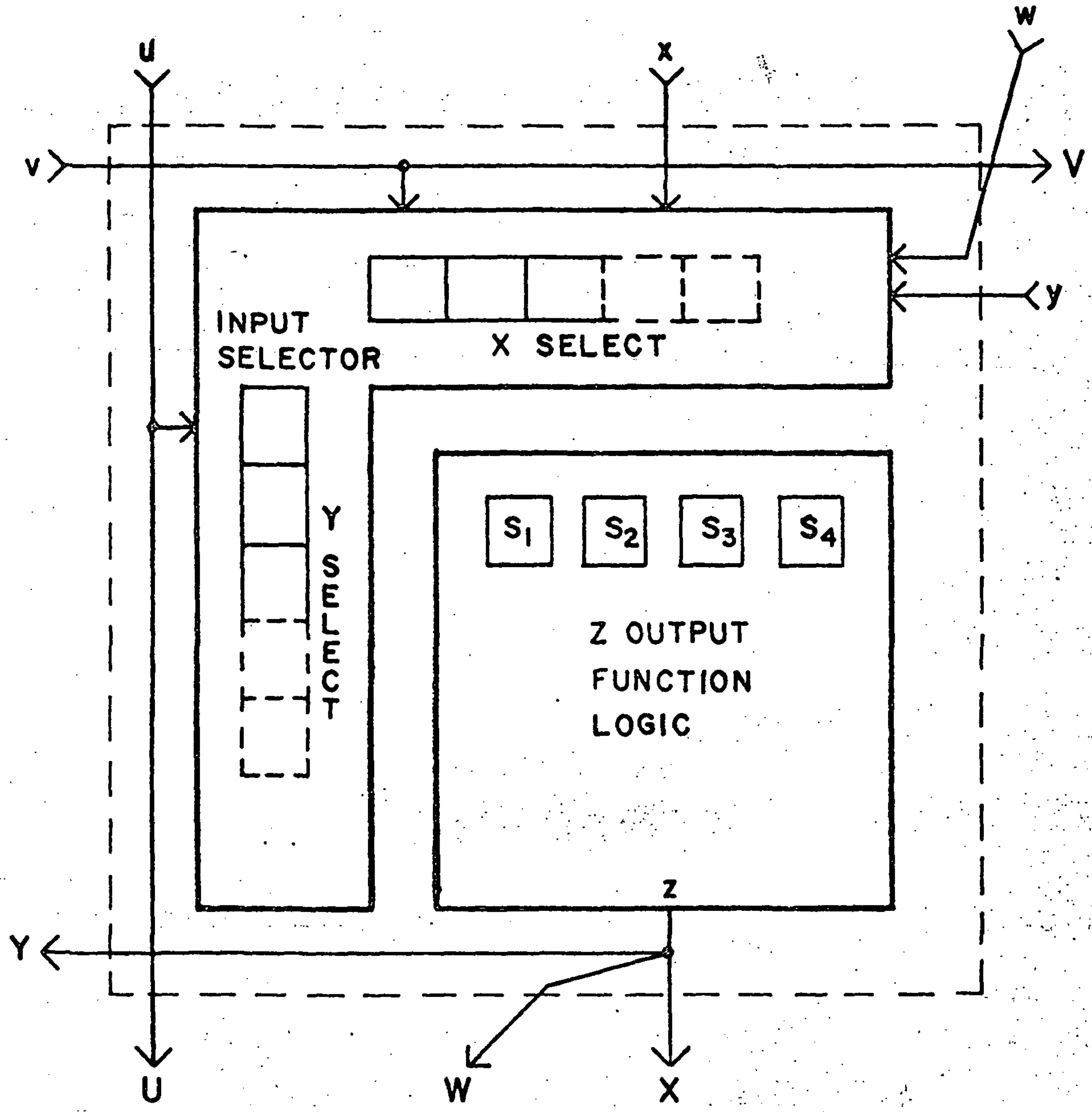


Fig. 21. Block form of one cobweb cell.

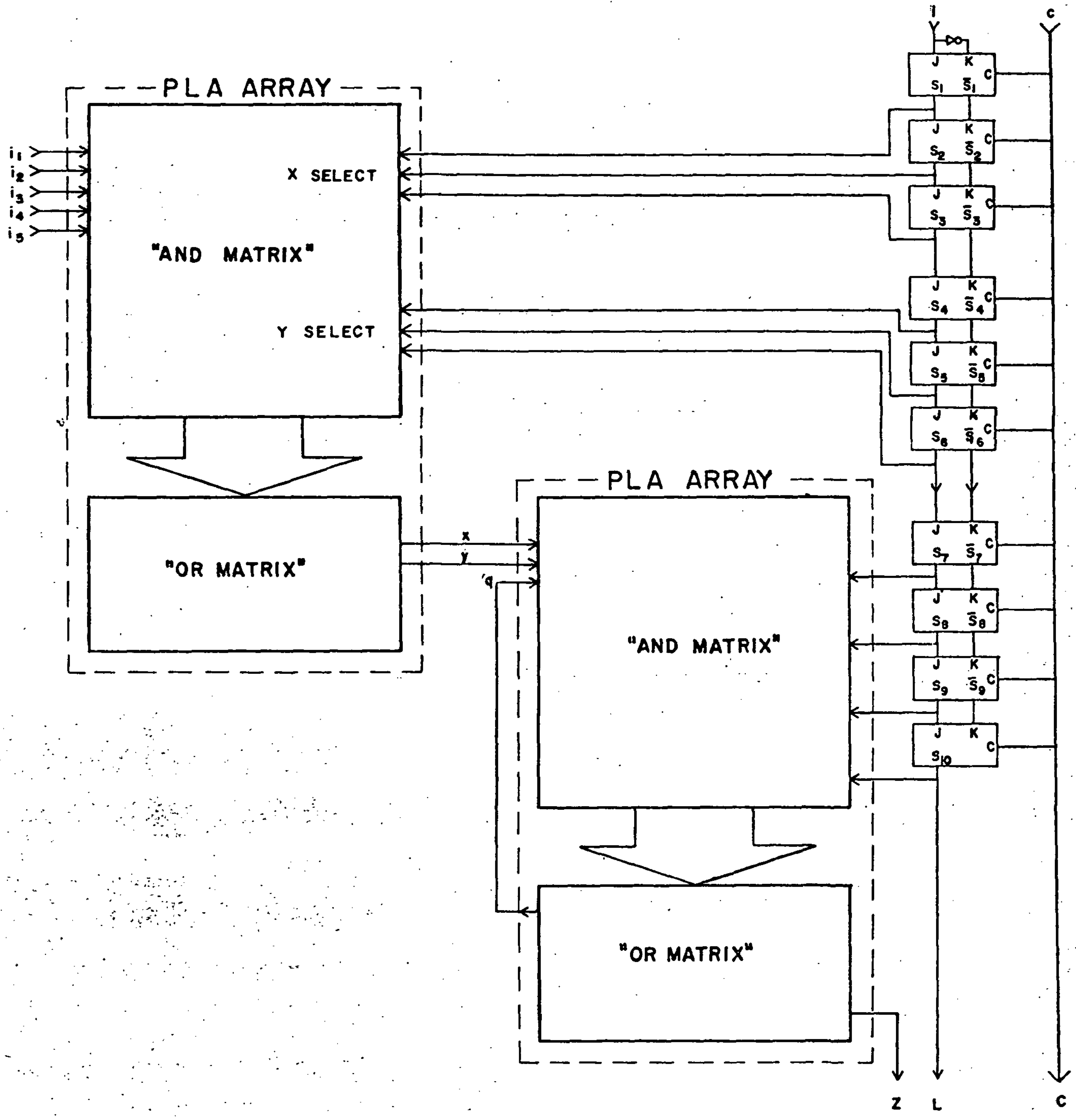


FIG 22. PLA IMPLEMENTATION OF THE COBWEB CELL.

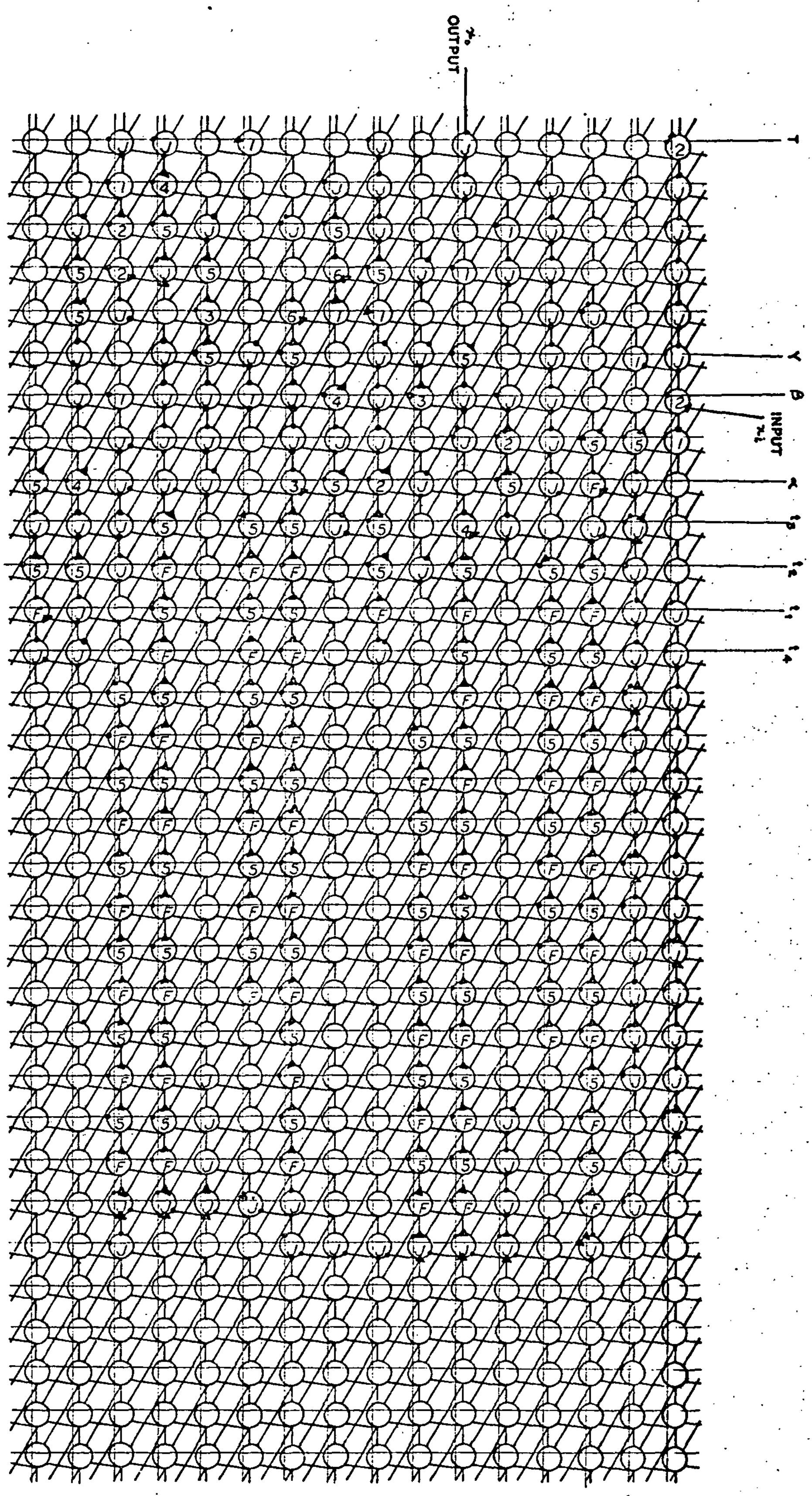


Fig. 23. Cobweb realization of a seven-bit serial multiplier.

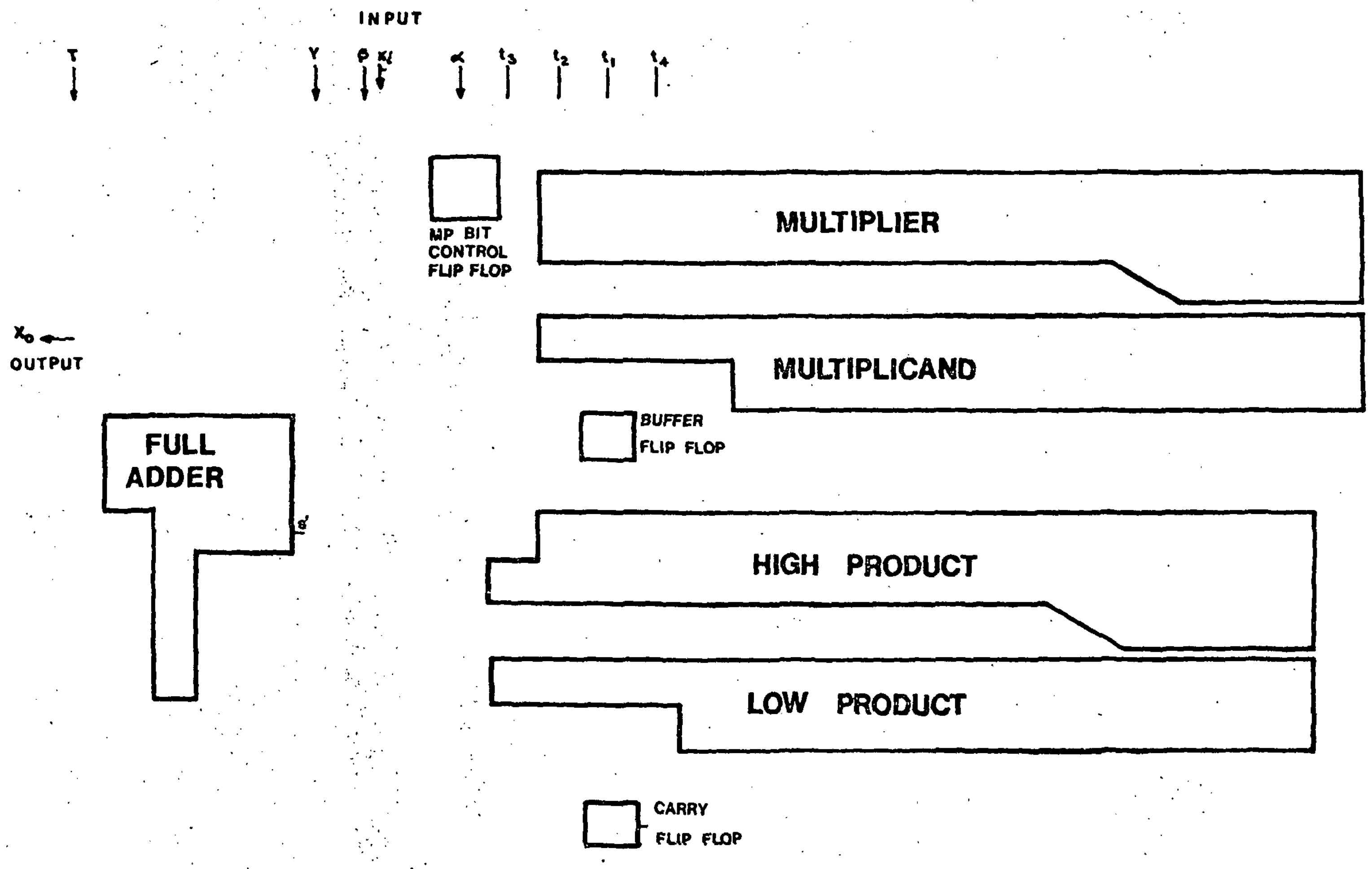


FIG. 24. OVERLAY FOR THE COBWEB MULTIPLIER.

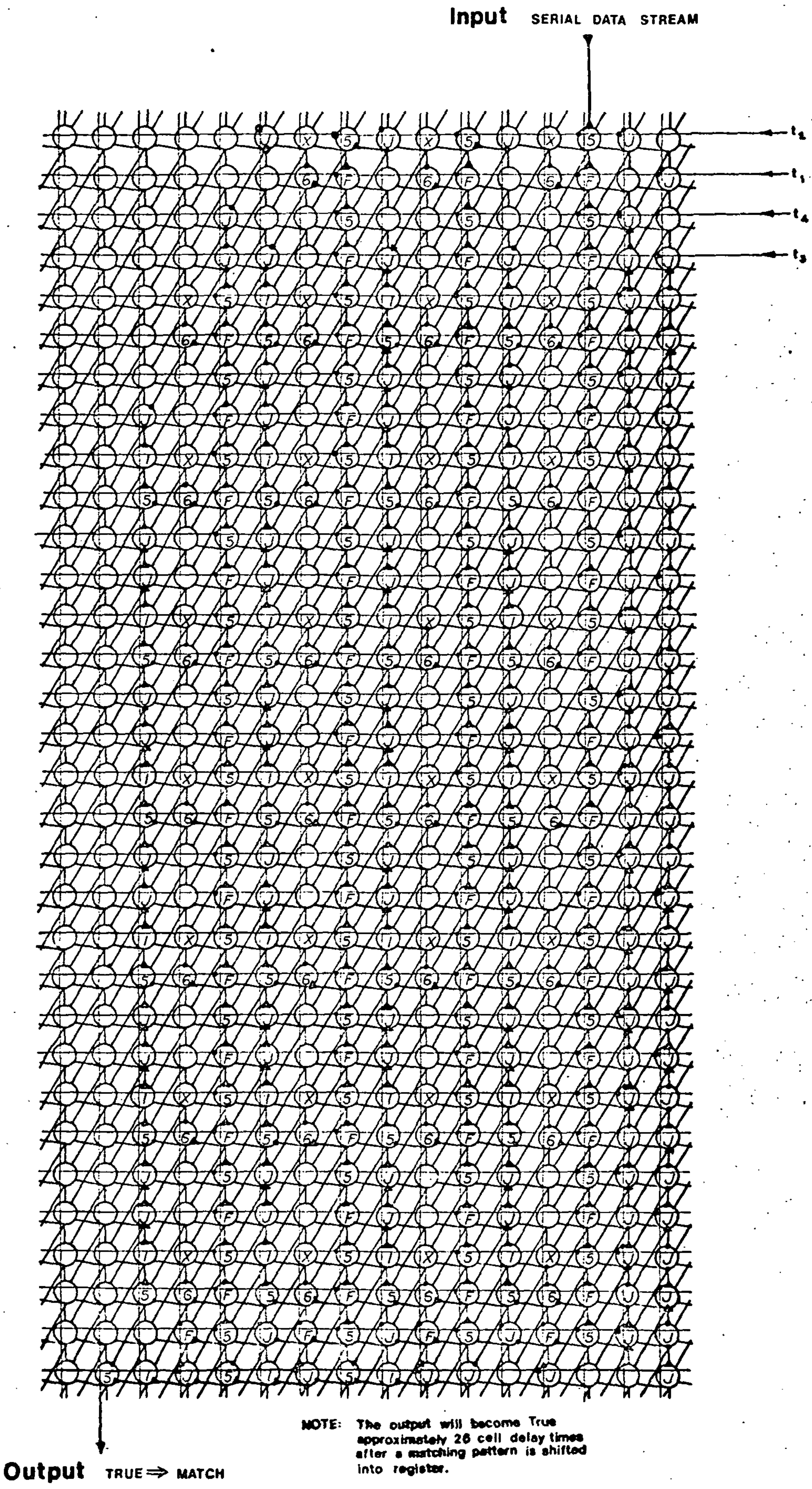

Input SERIAL DATA STREAM
t_2
t_1
t_4
t_3
Output TRUE ⇒ MATCH
NOTE: The output will become True approximately 26 cell delay times after a matching pattern is shifted into register.

FIG. 25. 31-BIT SEQUENCE DETECTOR.

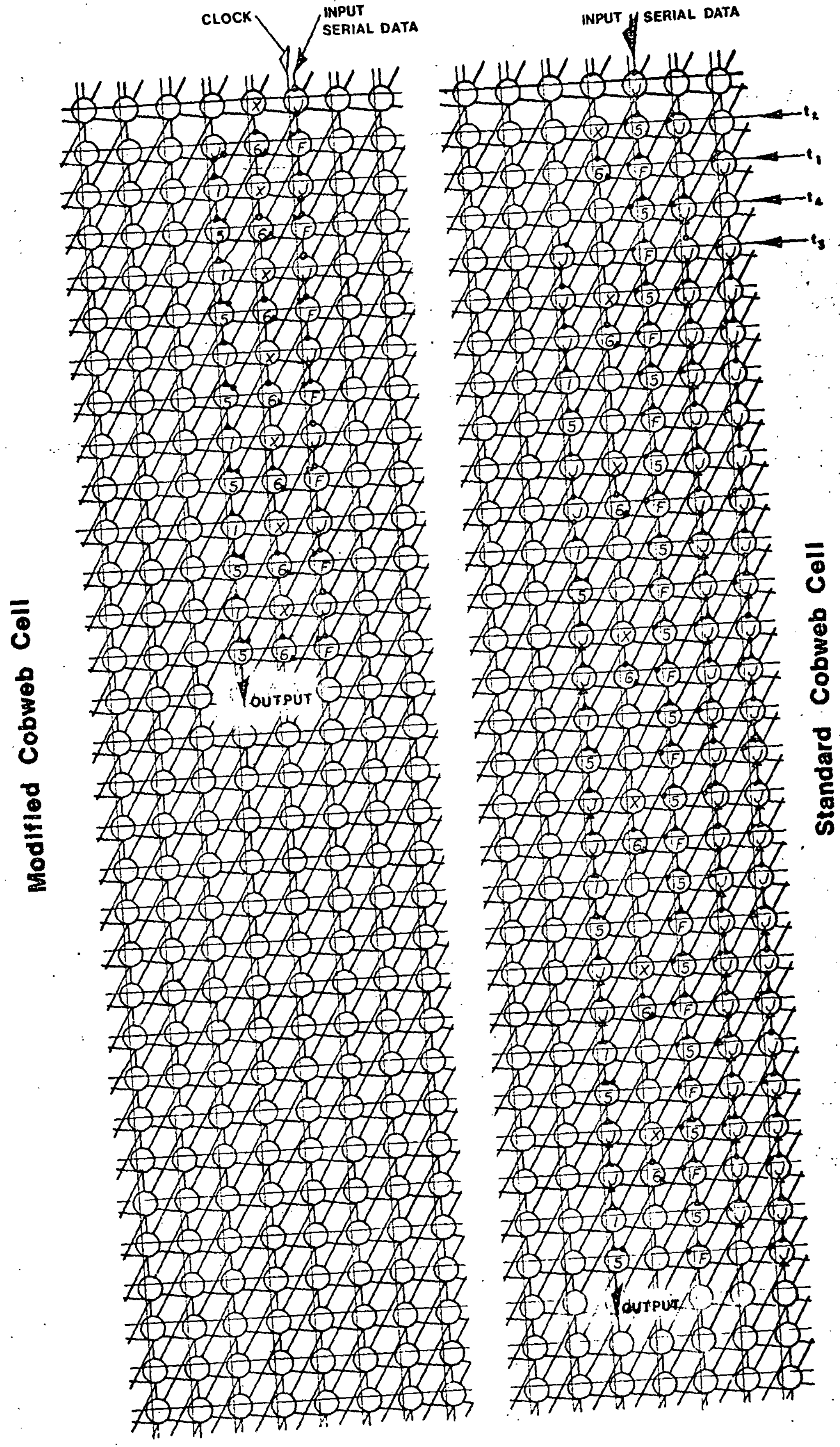


FIG. 26. SEVEN-BIT SEQUENCE DETECTOR USING TWO COBWEB CELLS.

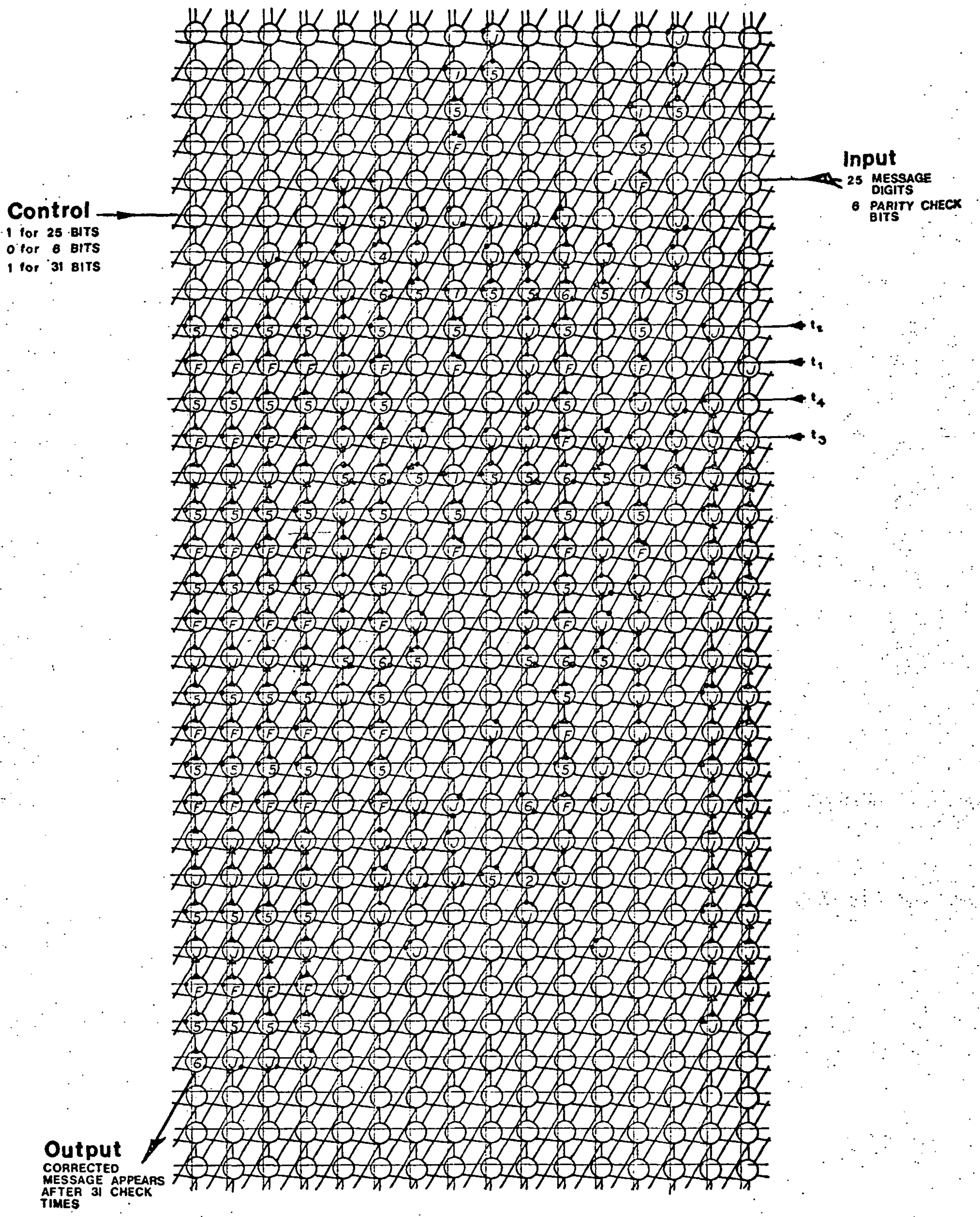


FIG. 27. COBWEB REALIZATION FOR A 31-BIT FEEDBACK SHIFT REGISTER DECODER.

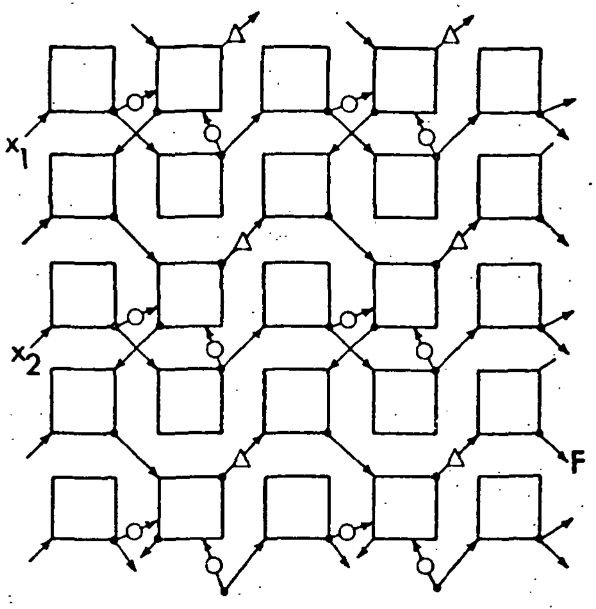


Fig. 28. Spandorfer array.

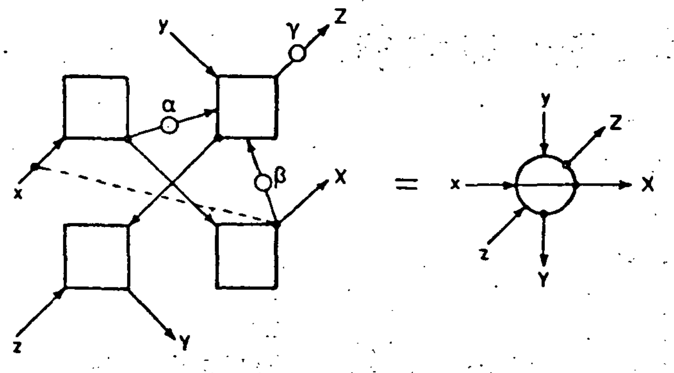
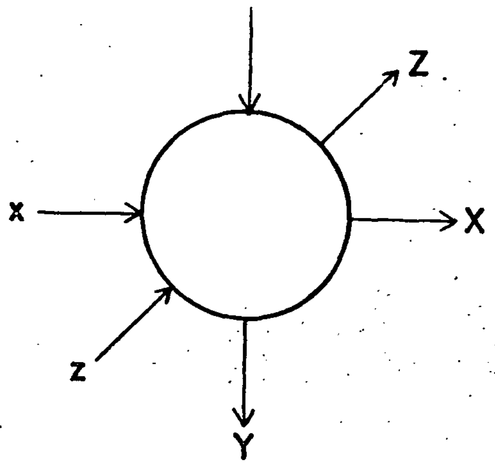
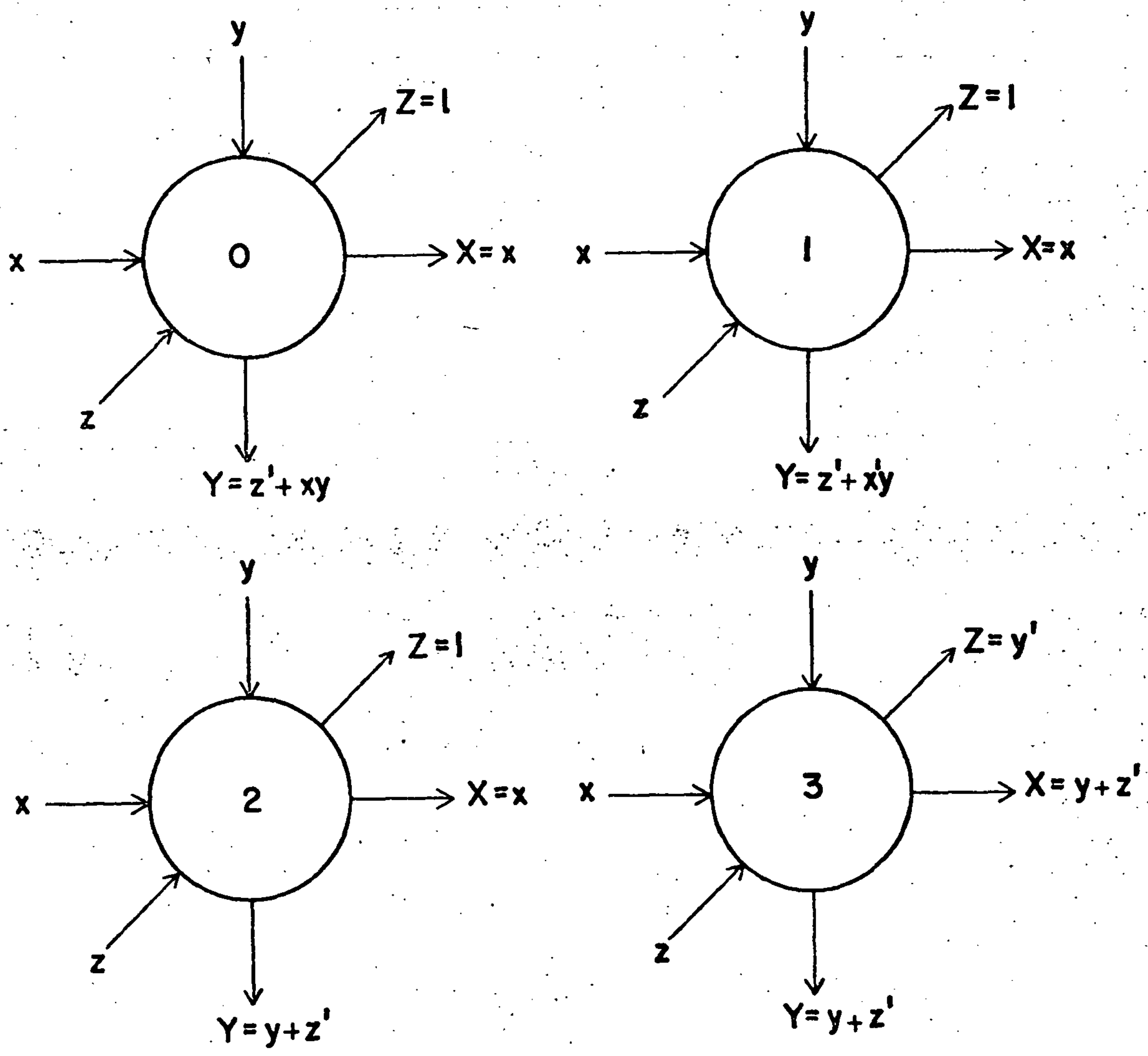


Fig. 29. Original programmable array cell.



(A) NOTATION FOR THE CELL.



(B) SPECIALIZATION.

FIG. 30. MODIFIED PROGRAMMABLE ARRAY CELL.

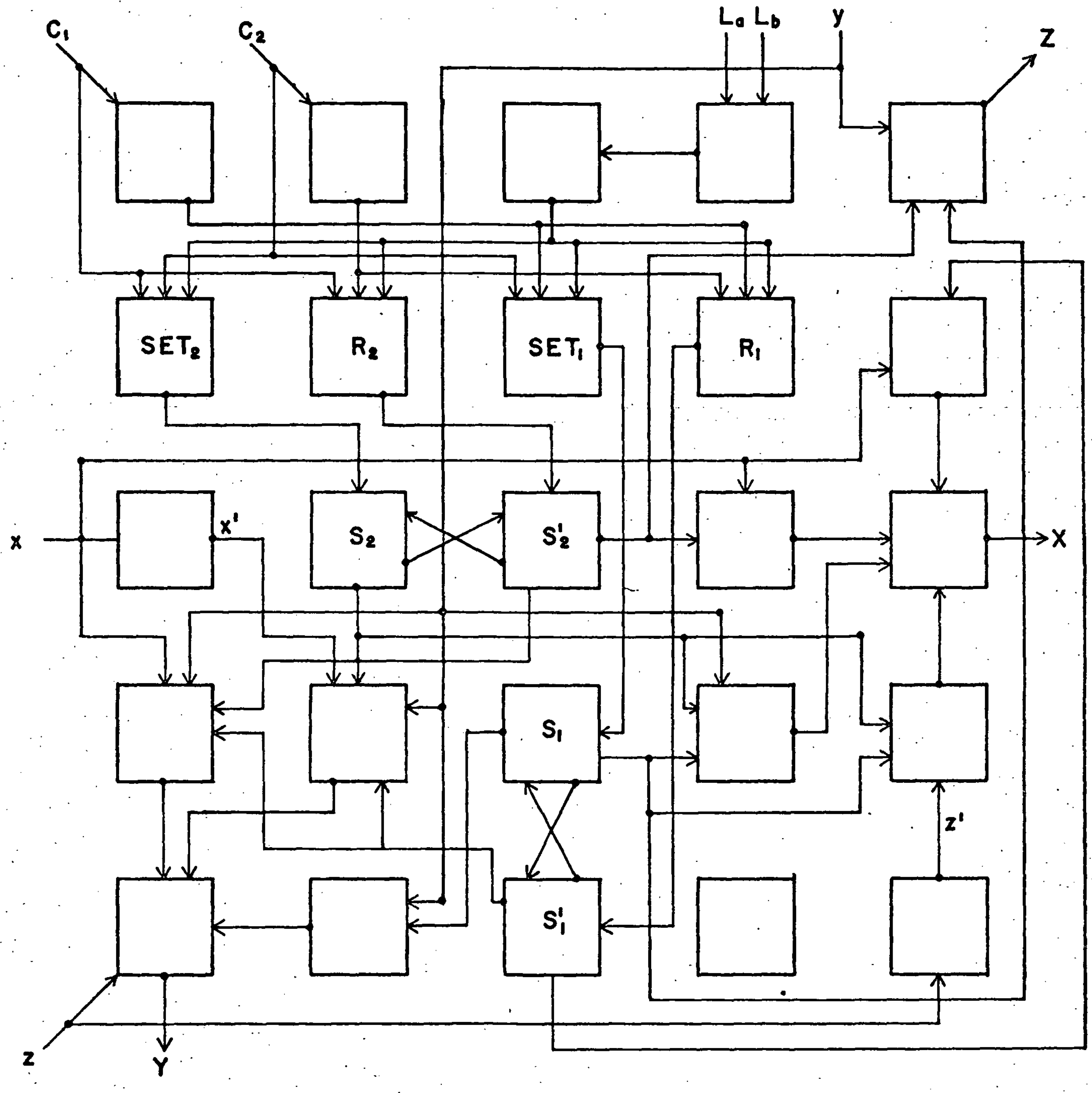


FIG. 31A. NAND REALIZATION OF THE MODIFIED PROGRAMMABLE CELL.

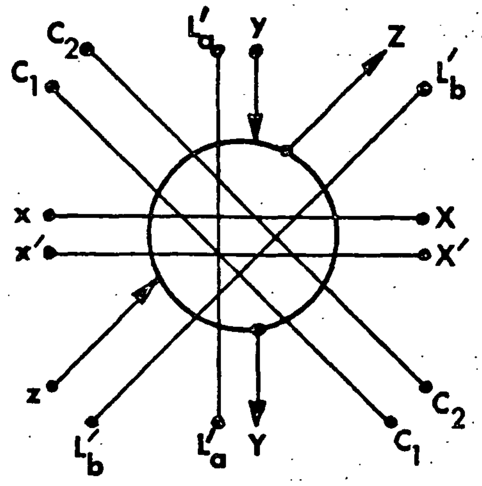


Fig. 31b. Notation for the modified programmable cell.

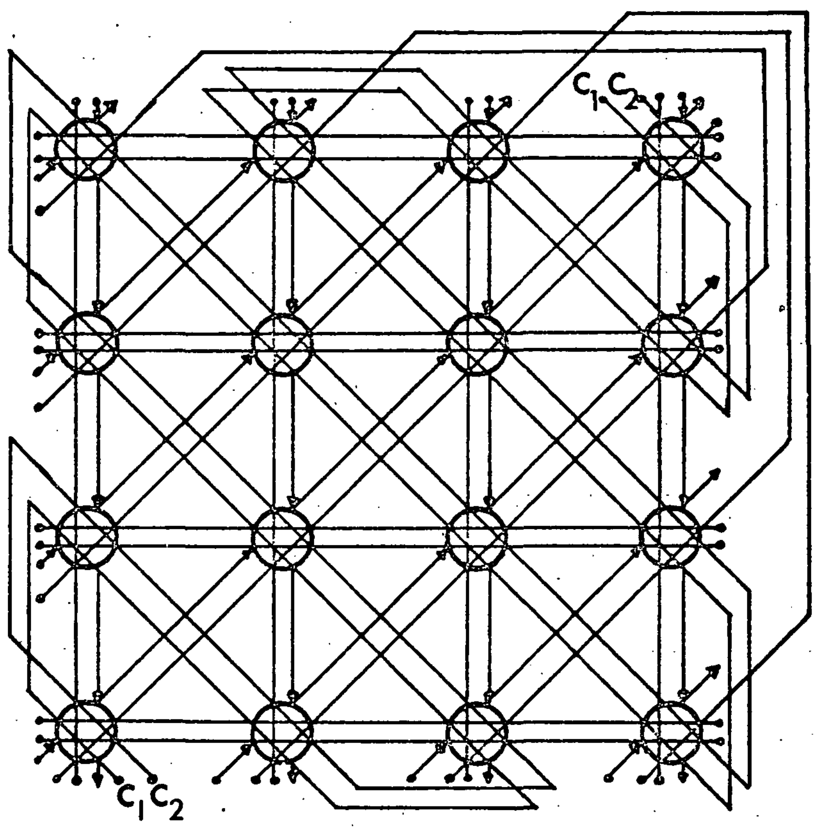


Fig. 32. Overall organization of the modified programmable array.

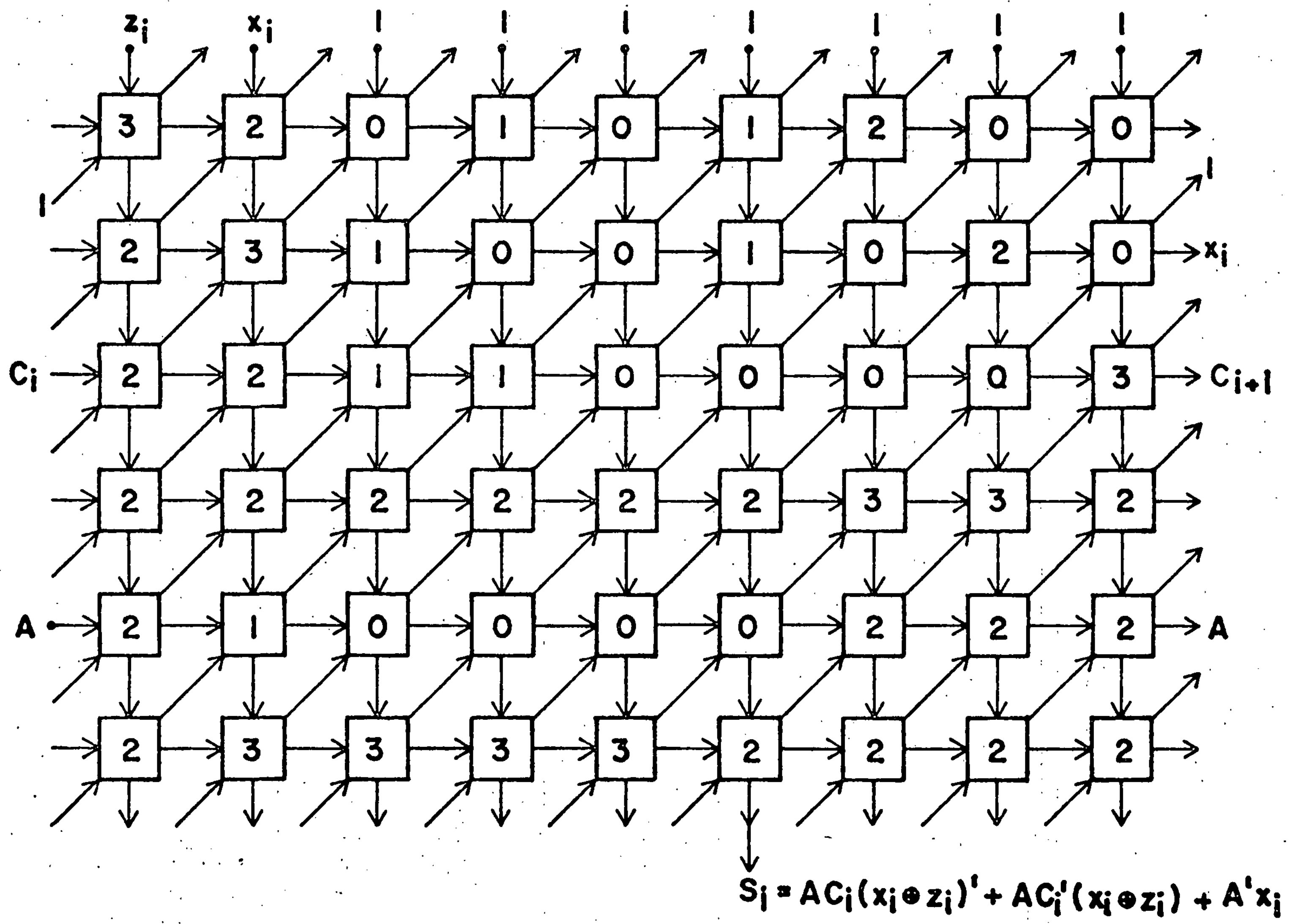


FIG. 33. GATED FULL ADDER FOR ONE BIT.

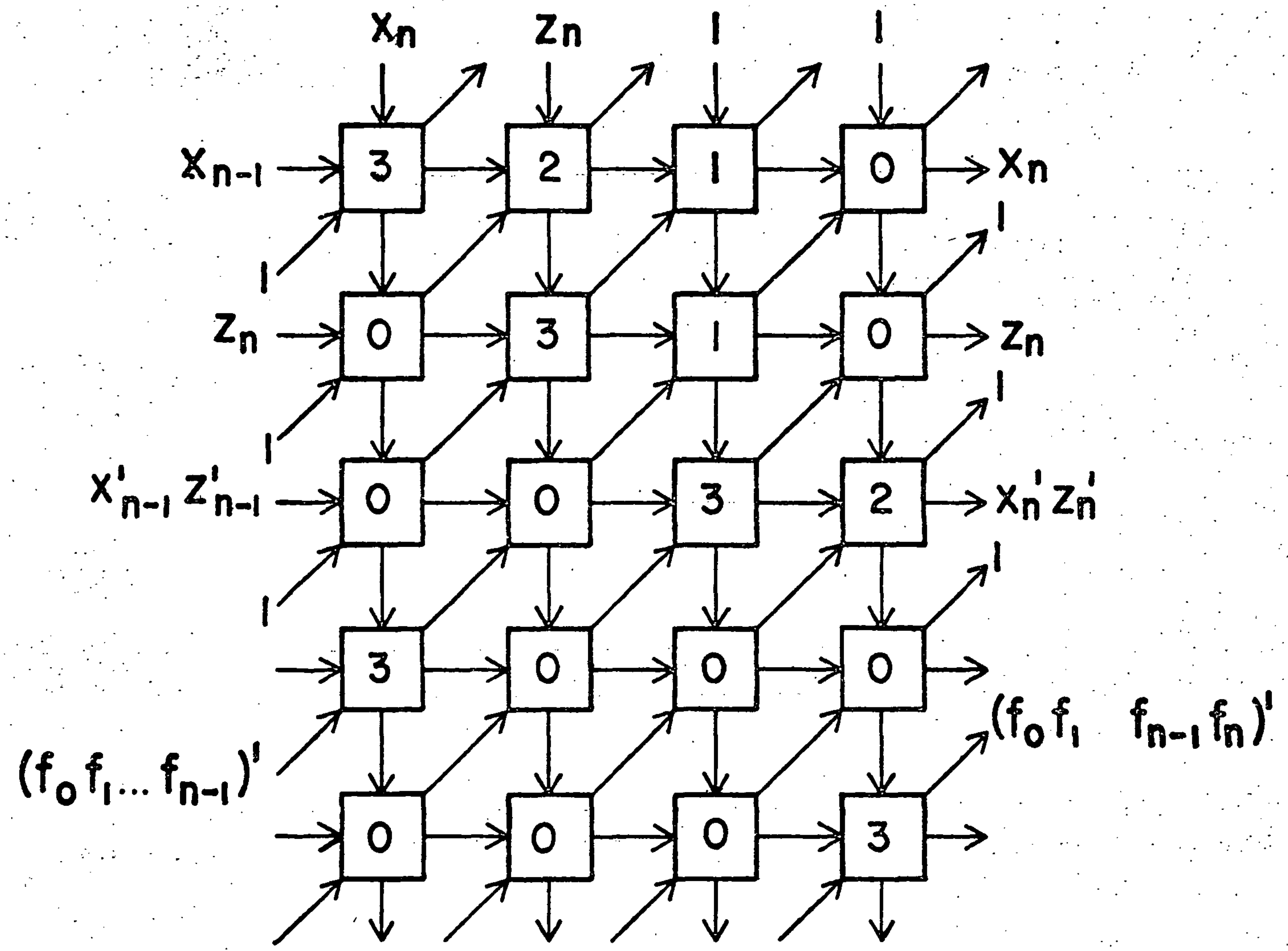


FIG. 34. ONE STAGE OF A SEQUENCE DETECTOR.

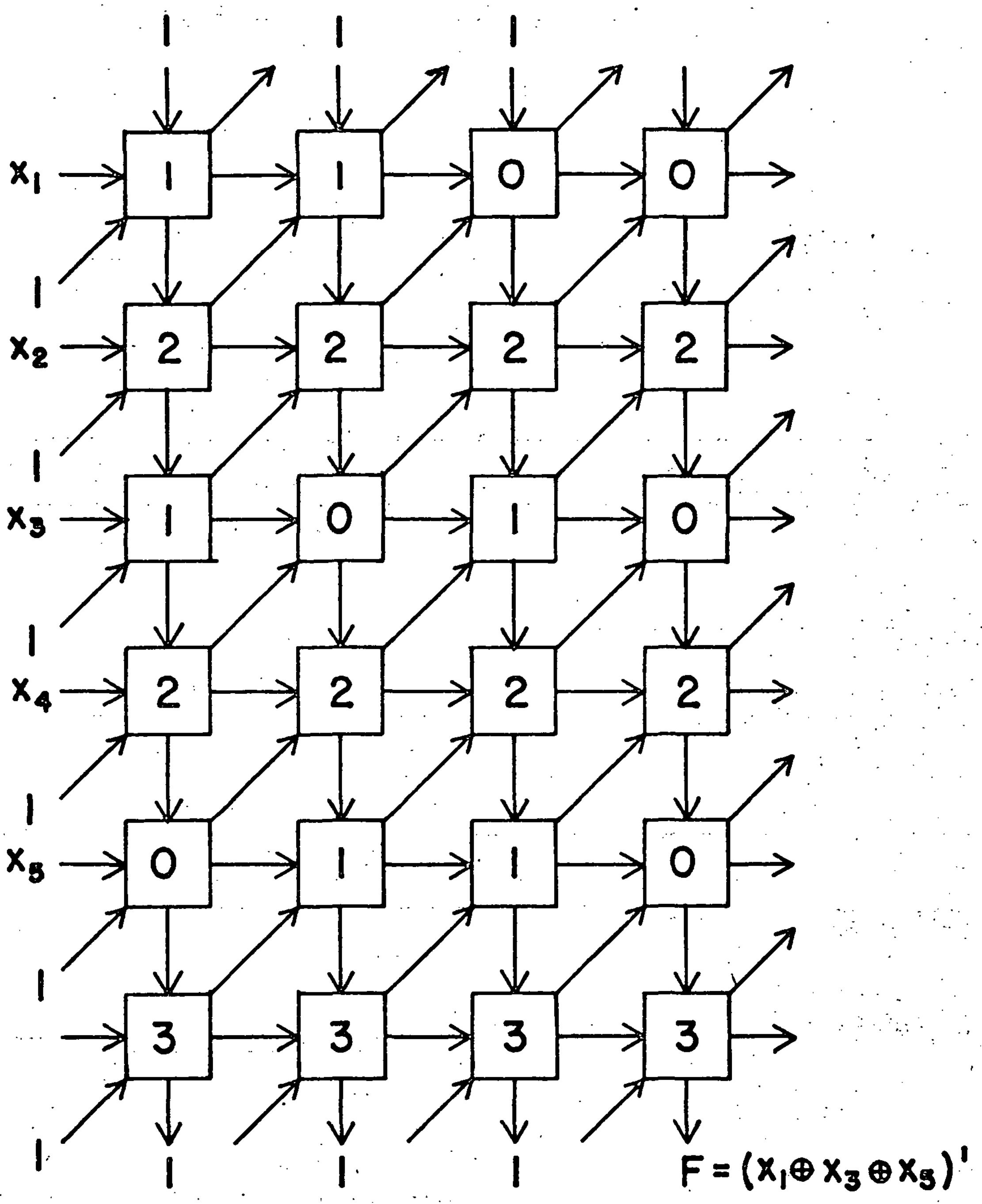


FIG. 35. SAMPLE LOGIC FOR FEEDBACK SHIFT REGISTER DECODER.